Randomized Half-Ideal Cipher on Groups with applications to UC (a)PAKE

Bruno Freitas Dos Santos[0009−0009−5474−0008], Yanqi Gu[0000−0001−6577−2704], and Stanislaw Jarecki[0000−0002−5055−2407]

University of California, Irvine. Email: {brunof,yanqig1,sjarecki}@uci.edu

Abstract. An Ideal Cipher (IC) is a cipher where each key defines a random permutation on the domain. Ideal Cipher on a group has many attractive applications, e.g., the *Encrypted Key Exchange* (EKE) protocol for Password Authenticated Key Exchange (PAKE) [8], or asymmetric PAKE (aPAKE) [33, 31]. However, known constructions for IC on a group domain all have drawbacks, including key leakage from timing information [12], requiring 4 hash-onto-group operations if IC is an 8-round Feistel [22], and limiting the domain to half the group [9] or using variable-time encoding [47, 39] if IC is implemented via (quasi-) bijections from groups to bitstrings [33].

We propose an IC relaxation called a *(Randomized) Half-Ideal Cipher* (HIC), and we show that HIC on a group can be realized by a *modified 2-round Feistel* (m2F), at a cost of 1 hash-onto-group operation, which beats existing IC constructions in versatility and computational cost. HIC weakens IC properties by letting part of the ciphertext be non-random, but we exemplify that it can be used as a drop-in replacement for IC by showing that EKE [8] and aPAKE of [33] realize respectively UC PAKE and UC aPAKE even if they use HIC instead of IC. The m2F construction can also serve as IC domain extension, because m2F constructs HIC on domain D from an RO-indifferentiable hash onto D and an IC on 2κ-bit strings, for κ a security parameter. One application of such extender is a modular lattice-based UC PAKE using EKE instantiated with HIC and anonymous lattice-based KEM.

1 Introduction

The Ideal Cipher Model (ICM) dates back to the work of Shannon [46], and it models a block cipher as an Ideal Cipher (IC) oracle, where every key, even chosen by the attacker, defines an independent random permutation. Formally, an efficient adversary who evaluates a block cipher on any key k of its choice cannot distinguish computing the cipher on that key in the forward and backward direction from an interaction with oracles $E_k(\cdot)$ and $E_k^{-1}(\cdot)$, where $\{E_i\}$ is a family of random permutations on the cipher domain. The Ideal Cipher Model has seen a variety of applications in cryptographic analysis, e.g. [48, 44, 28, 45, 38, 24, 13, 37], e.g. the analysis of the Davies-Meyer construction of a collision-resistant hash [45, 13], of the Even-Mansour construction of a cipher from a

public pseudorandom permutation [28], or of the DESX method for key-length extension for block ciphers [38]. A series of works [26, 18, 36, 19, 22] shows that ICM is equivalent to the Random Oracle Model (ROM) [7]. Specifically, these papers show that n-round Feistel, where each round function is a Random Oracle (RO), implements IC for some n, and the result of Dai and Steinberger [22] shows that $n = 8$ is both sufficient and necessary. Other IC constructions include iterated Even-Mansour and key alternating ciphers [21, 4, 27], wide-input (public) random permutations [11, 10, 20], and domain extension mechanisms, e.g. [17, 34], constructions based on

Ideal Ciphers on Groups: Applications. All the IC applications above consider IC on a domain of fixed-length bitstrings. However, there are also attractive applications of IC whose domain is a *group*. A prominent example is a Password Authenticated Key Exchange (PAKE) protocol called *Encrypted Key Exchange* (EKE), due to Bellovin and Meritt [8]. EKE is a compiler from plain key exchange (KE) whose messages are pseudorandom in some domain D, and it implements a secure PAKE if parties use an IC on domain D to password-encrypt KE messages.[1] The EKE solution to PAKE is attractive because it realizes UC PAKE given any key-private (a.k.a. anonymous) KEM [5], or KE with a mild "random message" property, at a cost which is the same as the underlying KE(M) *if* the cost of IC on KE(M) message domain(s) is negligible compared to the cost of KE(M) itself. However, instantiating EKE with e.g. Diffie-Hellman KE (DH-KE) [25] requires an IC on a group because DH-KE messages are random group elements.

Recently Gu et al. [33] and Freitas et al. [31] extended the EKE paradigm to cost-minimal compilers which create UC *asymmetric* PAKE (aPAKE), i.e. PAKE for the client-server setting where one party holds a one-way hash of the password instead of a password itself, from any key-hiding Authenticated Key Exchange (AKE). The AKE-to-aPAKE compilers of [33, 31] are similar to the "EKE" KE-to-PAKE compiler of [8] in that they also require IC-encryption of KE-related values, but they use IC to password-encrypt a KEM public key rather than KE protocol messages. The key-hiding AKE's exemplified in [33, 31], namely HMQV [40] and 3DH [42], are variants and generalizations of DH-KE where public keys are group elements, hence the AKE-to-aPAKE compilers of [33, 31] instantiated this way also require IC on a group.

Ideal Ciphers on Groups: Existing Constructions. The above motivates searching for efficient constructions of IC on a domain of an arbitrary group. Note first that a standard block cipher on a bitstring domain does not work. The elements of any group G can be encoded as bitstrings of some fixed length n, but unless these encodings cover almost all n-bit strings, i.e. unless $(1-|G|/2^n)$ is negligible, encrypting G elements under a password using IC on n-bit strings

[1] Bellare et al. [6] showed that EKE+IC is a game-based secure PAKE, then Abdalla et al. [1] showed that EKE variant with explicit key confirmation realizes UC PAKE, and recently McQuoid et al. [43] showed that a round-minimal EKE variant realizes UC PAKE as well (however, see more on their analysis below).

exposes a scheme to an offline dictionary attack, because the adversary can decrypt a ciphertext under any password candidate and test if the decrypted plaintext encodes a G element.

Black and Rogaway [12] showed an elegant black-box solution for an IC on G given an IC on n-bit strings provided that $c = (2^n/|G|)$ is a constant: To encrypt element $x \in G$ under key k, use the underlying n-bit IC in a loop, i.e. set x_0 to the n-bit encoding of x, and $x_{i+1} = \mathsf{IC.Enc}_k(x_i)$ for each $i \geq 0$, and output as the ciphertext the first x_i for $i \geq 1$ s.t. x_i encodes an element of group G. (Decryption works the same way but using $\mathsf{IC.Dec}$.) This procedure takes expected c uses of $\mathsf{IC.Enc}$, but timing measurement of either encryption or decryption leaks roughly $\log c$ bits of information on key k per each usage, because given the ciphertext one can eliminate all keys which form decryption cycles whose length does not match the length implied by the timing data.

To the best of our knowledge there are only two other types of constructions of IC on a group. First, the work of [26, 18, 36, 19, 22] shows that n-round Feistel network implements an IC for $n \geq 8$. Although not stated explicitly, these results imply a (randomized) IC on a group, where one Feistel wire holds group elements, the xor gates on that wire are replaced by group operations, and hashes onto that wire are implemented as RO hashes onto the group. However, since $n = 8$ rounds is minimal [22], this construction incurs four RO hashes onto a group per cipher operation. Whereas there is progress regarding RO-indifferentiable hashing on Elliptic Curve (EC) groups, see e.g. [29], current implementations report an RO hash costs in the ballpark of 25% of scalar multiplication. Hence, far from being negligible, the cost of IC on group implemented in this way would roughly equal the DH-KE cost in the EKE compiler. The second construction of (randomized) IC combines any (randomized) quasi-bijective encoding of group elements as bitstrings with an IC on the resulting bitstrings [33]. However, we know of only two quasi-bijective encodings for Elliptic Curve groups, Elligator2 of Bernstein et al. [9] and Elligator^2 of Tibouchi et al. [47, 39], and both have some practical disadvantages. Elligator2 works for only some elliptic curves, and it can encode only half the group elements, which means that any application has to re-generate group elements until it finds one in the domain of Elligator2. Elligator^2 works for a larger class of curves, but its encoding procedure is non-constant time and it appears to be significantly more expensive than one RO hash onto a curve. Elligator^2 also encodes each EC element as a pair of underlying field elements, effectively doubling the size of the EC element representation.

IC Alternative: Programmable-Once Public Function. An alternative path was recently charted by McQuoid et al. [43], who showed that a 2-round Feistel, with one wire holding group elements, implements a randomized cipher on a group which has some IC-like properties, which [43] captured in a notion of Programmable Once Public Function (POPF). Moreover, they argue that POPF can replace IC in several applications, exemplifying it with an argument that EKE realizes UC PAKE if password encryption is implemented with a POPF in place of IC. This would be very attractive because if 2-round Feistel can indeed function as an IC replacement in applications like the PAKE of

[8] or the aPAKE's of [33, 31], this would form the most efficient and flexible implementation option for these protocols, because it works for any group which admits RO-indifferentiable hash, and it uses just one such hash-onto-group per cipher operation.

However, it seems difficult to use the POPF abstraction of [43] as a replacement for IC in the above applications because the POPF notion captures 2-round Feistel properties with game-based properties which appear not to address *non-malleability*. For that reason we doubt that it can be proven that UC PAKE is realized by EKE with IC replaced by POPF as defined in [43]. (See below for more details.) The fact that the POPF abstraction appears insufficient does not preclude that UC PAKE can be realized by EKE with encryption implemented as 2-round Feistel, but such argument would not be modular. Moreover, each application which uses 2-round Feistel in place of IC would require a separate non-modular proof. Alternatively, one could search for a "POPF+" abstraction, realized by a 2-round Feistel, which captures sufficient non-malleability properties to be useful as an IC replacement in PAKE applications, but in this work we chose a different route.

Our Results: Modified 2-Feistel as (Randomized) Half-Ideal Cipher. Instead of trying to work with 2-Feistel itself, we show that adding a block cipher BC to one wire in 2-Feistel makes this transformation non-malleable, and we capture the properties of this construction in the form of a UC notion we call a (Randomized) Half-Ideal Cipher (HIC). In Figure 1 we show a simple pictorial comparison of 2-Feistel, denoted $\mathsf{2F}$, and our modification, denoted $\mathsf{m2F}$. The modified 2-Feistel has the same efficiency and versatility as the 2-Feistel used by McQuoid et al. [43]: It works for any group with an RO-indifferentiable hash onto a group, it runs in fixed time, and it requires only one RO hash onto a group per cipher operation.

One drawback of $\mathsf{m2F}$ is that the ciphertext is longer than the plaintext by 2κ bits, where κ is a security parameter. However, that is less than any IC implementation above (including POPF, which does not realize IC) except for Elligator2: IC results from n-round Feistel have loose security bounds, hence they need significantly longer randomness to achieve the same provable security; Elligator2 adds κ bits for general moduli, due to encoding of field elements as random bitstrings; Elligator^2 uses an additional field element, which adds at least 2κ bits, plus another κ bits for the field-onto-bits encoding; Finally, 2-Feistel requires at least 3κ bits of randomness when used in EKE [43].

The UC HIC notion is a relaxation of an Ideal Cipher notion, but it does not prevent applicability in protocols like [8, 33, 31], which we exemplify by showing that the following protocols remain secure with (any realization of) IC replaced by (any realization of) HIC:

(I) UC PAKE is realized by an EKE variant with IC replaced by HIC, using round-minimal KE with a random-message property;
(II) UC PAKE is realized by an EKE variant with IC replaced by HIC, using anonymous KEM with a uniform public keys property;

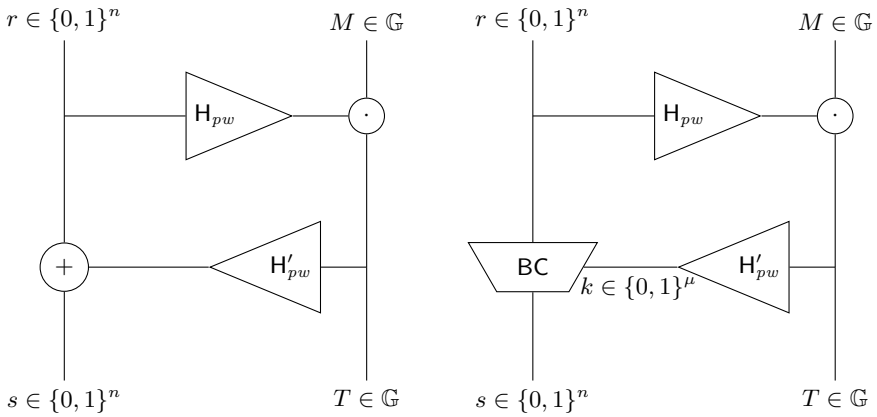

Fig. 1: Left: two-round Feistel (2F) used in McQuoid et al. [43]; Right: our circuit m2F. The change from 2F to m2F is small: If $k = H'(pw, T)$, then 2F sets $s = k \oplus r$, whereas m2F sets $s = \mathsf{BC.Enc}(k, r)$, where BC is a block cipher.

(III) UC aPAKE is realized by KHAPE [33] with IC replaced by HIC, using key-hiding AKE.

Regarding the first two proofs, we are not aware of full proofs exhibited for the corresponding statements where these EKE variants use IC instead of HIC, but the third proof follows the blueprint of the proof given in [33] for the KHAPE protocol using IC, and it exemplifies how little such proof changes if IC is replaced by HIC.

Half-Ideal Cipher. The first difference between IC on group $\mathbb{G}$ and HIC on group $\mathbb{G}$ is that the latter is a cipher on an extended domain $\mathcal{D} = \mathcal{R} \times \mathbb{G}$ where $\mathcal{R} = \{0,1\}^n$ is the randomness space, for $n \geq 2\kappa$ where κ is the security parameter. In the decryption direction, HIC acts exactly like IC on domain $\mathcal{D}$, i.e. unless ciphertext $c \in \mathcal{D}$ is already associated with some plaintext in the permutation table defined by key k, an adversarial decryption of c under key k returns a random plaintext m, chosen by the HIC functionality with uniform distribution over those elements in domain $\mathcal{D}$ which are not yet assigned to any ciphertext in the permutation table for key k. However, in the encryption direction HIC is only *half-ideal* in the following sense: If plaintext m is not yet associated with any ciphertext in the permutation table for key k then encryption of m under key k returns a ciphertext $c = (s, T) \in \mathcal{D} = \mathcal{R} \times \mathbb{G}$ s.t. the $T \in \mathbb{G}$ part of c *can be freely specified by the adversary*, and the $s \in \mathcal{R}$ part of c is then chosen by the HIC functionality at random with uniform distribution over s's s.t. $c = (s, T)$ is not yet assigned to any plaintext in the permutation table for key k. In short, HIC decryption on any (k, c) returns a random plaintext m (subject to the constraint that $\mathsf{HIC}(k, \cdot)$ is a permutation on $\mathcal{D}$), but HIC encryption on any (k, m) returns $c = (s, T)$ s.t. T can be correlated with other values in an

arbitrary way, which is modeled by allowing the adversary to choose it, but s is random (subject to the constraint that $\mathsf{HIC}(k, \cdot)$ is a permutation).[2]

Intuitively, the reason the adversarial ability to manipulate part of IC ciphertext does not affect typical IC applications is that these applications typically rely on the following properties of IC: (1) that decryption of a ciphertext on any other key from the one used in encryption outputs a random plaintext, (2) that any change to a ciphertext implies that the corresponding plaintext is random and hence uncorrelated to the plaintext in the original ciphertext, and (3) that no two encryption operations can output the same ciphertext, regardless of the keys used, and moreover that the simulator can straight-line extract the unique key used in a ciphertext formed in the forward direction. Only properties (2) and (3) could be affected by the adversarial ability to choose the T part of a ciphertext in encryption, but the fact that the s part is still random, and that $|s| \geq 2\kappa$, means that just like in IC, except for negligible probability each encryption outputs a ciphertext which is different from all previously used ones. Consequently, just like in IC, a HIC ciphertext commits the adversary to (at most) a *single* key used to create that ciphertext in a forward direction, the simulator can straight-line extract that key, and the decryption of this ciphertext under any other key samples random elements in the domain.

Further Applications: IC domain extension, LWE-based UC PAKE. The modified 2-Feistel construction can also be used as a *domain extender* for (randomized) IC on *bitstrings*. Given an RO hash onto $\{0,1\}^t$ and an IC on $\{0,1\}^{2\kappa}$, the $\mathsf{m2F}$ construction creates a HIC on $\{0,1\}^t$, for any $t = \mathsf{poly}(\kappa)$. The modified 2-Feistel is simpler than other IC domain extenders, e.g. [17, 34], and it has better exact security bounds, hence it is an attractive alternative in applications where HIC can securely substitute for IC on a large bitstring domain. For example, by our result (II) above, $\mathsf{m2F}$ on long bitstrings can be used to implement UC PAKE from any lattice-based IND-secure and anonymous KEM. This includes several post-quantum LWE-based KEM proposals in the NIST competition, including Saber [23], Kyber [14], McEliece [2], NTRU [35], Frodo [3], and possibly others.[3] Such UC PAKE construction would add only 3κ bits in bandwidth to the underlying KEM, and its computational overhead over the underlying KEM operations would be negligible, i.e. the LWE-based UC PAKE would have essentially exactly the same cost as the LWE-based unauthenticated Key Exchange, i.e. an

[2] This describes only the *adversarial* interface to the HIC functionality. Honest parties' interface is as in IC in both directions, except that it hides encryption randomness, i.e. encryption takes only input $M \in \mathbb{G}$ and decryption outputs only the $M \in \mathbb{G}$ part of the "extended" HIC plaintext $m \in \mathcal{D}$.

[3] Two recent papers [41, 49] investigate anonymity of several CCA-secure LWE-based KEMs achieved via variants of the Fujisaki-Okamoto transform [32] applied to the IND-secure versions of these KEM's. However, the underlying IND-secure KEM's are all anonymous, see e.g. [41, 49] and the references therein.

IND-secure KEM. We show a concrete construction of UC PAKE from Saber KEM in the full version [30].

Half-Ideal Cipher versus POPF. Our modified 2-Feistel construction and the UC HIC abstraction we use to capture its properties can be thought of as a "non-malleability upgrade" to the 2-Feistel, and to the game-based POPF abstraction used by McQuoid et al. [43] to capture its properties. One reason why the UC HIC notion is an improvement over the POPF notion is that a UC tool is easier to use in protocol applications than a game-based abstraction. More specifically, the danger of game-based properties is that they often fail to adequately capture non-malleability properties needed in protocol applications, e.g. in the EKE protocol, where the man-in-the-middle attacker can modify the ciphertexts exchanged between Alice and Bob.[4] Indeed, POPF properties seem not to capture ciphertext non-malleability. As defined in [43], POPF has two security properties, *honest simulation* and *uncontrollable outputs*. The first one says that if ciphertext c is output by a simulator on behalf of an honest party, then decrypting it under any key results in a random element in group $\mathbb{G}$, except for the (key,plaintext) pair, denoted (x^*, y^*) in [43], which was programmed into this ciphertext by the simulator. The second property says that any ciphertext c^* output by an adversary decrypts to random elements in group $\mathbb{G}$ for all keys except for key k^*, denoted x^* in [43], which was used by the adversary to create c^* in the forward direction, and which can be straight-line extracted by the simulator.[5] However, these properties do not say that the (key,plaintext) pairs behind the adversary's ciphertext c^* cannot bear any relation to the (key,plaintext) pairs behind the simulator's ciphertext c.

Note that non-malleability is necessary in a protocol application like EKE, and for that reason we think that it is unlikely that EKE can provably realize UC PAKE based on the POPF properties alone. Consider a cipher Enc on a multiplicative group s.t. there is an efficient algorithm A s.t. if $c = \mathsf{Enc}(k, M)$ and $c^* = A(c)$ then $M^* = \mathsf{Dec}(k, c^*)$ satisfies relation $M^* = M^2$ if $\mathsf{lsb}(k) = 0$, and $m^* = m^3$ if $\mathsf{lsb}(k) = 1$. If this cipher is used in EKE for password-encryption of DH-KE messages then the attacker would learn lsb of password pw used by Alice and Bob: If the attacker passes Alice's message $c_A = \mathsf{Enc}(pw, g^x)$ to Bob, but replaces Bob's message $c_B = \mathsf{Enc}(pw, g^y)$ by sending a modified message $c_B^* = A(c_B)$ to Alice, then $c_B^* = \mathsf{Enc}(pw, g^{y \cdot (2+b)})$ where $b = \mathsf{lsb}(pw)$, hence an attacker who sees Alice's output $k_A = g^{xy \cdot (2+b)}$ and Bob's output $k_B = g^{xy}$, can learn bit b by testing if $k_A = (k_B)^{(2+b)}$. More generally, any attack A which transforms ciphertext $c = \mathsf{Enc}(k, M)$ to ciphertext $c^* = \mathsf{Enc}(k^*, M^*)$ s.t. (k, M, k^*, M^*) are in some non-trivial relation, is a potential danger for EKE.

[4] A potential benefit of a game-based notion over a UC notion is that the former *could* be easier to state and use, but this does not seem to be the case for the POPF properties of [43], because they are quite involved and subtle.

[5] Technically [43] state this property as pseudorandomness of outputs of any weak-PRF on the decryptions of c^* for any $k \neq k^*$, and not the pseudorandomness of the decrypted plaintexts themselves.

We do not believe that 2-Feistel is subject to such attacks, but POPF properties defined in [43] do not seem to forbid them.

If one uses 2-Feistel directly rather than the POPF abstraction then it might still be possible to prove that EKE with 2-Feistel realizes UC PAKE. We note that 2-Feistel is subject to the following restricted form of "key-dependent malleability", which appears not to have been observed in [43] and which would have to be accounted for in such proof. Namely, consider an adversary who given ciphertext $c = (s, T)$ outputs ciphertext $c^* = (s^*, T^*)$ for any T^* and s^* s.t. $s^* \oplus \mathsf{H}'(pw^*, T^*) = s \oplus \mathsf{H}'(pw^*, T)$. Note that this adversary is not performing a decryption of c under pw^*, because it is not querying $\mathsf{H}(pw^*, r)$ for $r = s \oplus \mathsf{H}'(pw^*, T)$, but plaintexts $M^* = \mathsf{Dec}(pw, c^*)$ and $M = \mathsf{Dec}(pw, c)$ satisfy a non-trivial relation $M^*/M = T^*/T$ if $pw = pw^*$ and not otherwise. On the other hand, since this adversarial behavior seems to implement just a different form of an online attack using a unique password guess pw^*, it is still possible that EKE realizes UC PAKE even when password encryption is implemented as 2-Feistel. However, rather than considering such non-modular direct proofs for each application of IC on a group, in this paper we show that a small change in the 2-Feistel circuit implies realizing a HIC relaxation of the IC model, and this HIC relaxation is as easy to use as IC in the security proofs for protocols like EKE [8] or aPAKE's of Gu et al. [33, 31].

Finally, we note that an extension of the above attack shows that 2-Feistel itself, without our modification, cannot realize the HIC abstraction. Observe that if the adversary computes t hashes $Z_i = \mathsf{H}(pw, r_i)$ for some pw and $r_1, ..., r_t$ and then t hashes $k_j = \mathsf{H}'(pw, T_j)$ for some $T_1, ..., T_t$, then it can combine them to form t^2 valid (plaintext, ciphertext) pairs (M_{ij}, c_{ij}) under key pw where $M_{ij} = Z_i \cdot T_j$ and $c_{ij} = (r_i \oplus k_j, T_j)$. Note that the t^2 plaintexts are formed using just $2t$ group elements $(Z_1, T_1), ..., (Z_t, T_t)$, so they are correlated. For example, the value of quotient $M_{ij}/M_{i'j}$ is the same for every j. Creating such correlations on plaintexts is impossible in the UC HIC, hence 2-Feistel by itself, without our modification, does not realize it.

Roadmap. In Section 2, we recall the syntax and properties of Key Exchange (KE) and Key Encapsulation Mechanism (KEM). In Section 3 we define the UC notion of Half-Ideal Cipher (HIC). In Section 4 we present the modified 2-Feistel construction, and we show that it realizes UC HIC. In Section 5 we define two variants of the EKE protocol, denoted EKE and EKE-KEM, based on respectively KE and KEM, with password encryption implemented as HIC, and we show that both variants realizes UC PAKE.

Because of space constraints we defer some parts to the full version of this paper [30]. Specifically, the full version contains the details of game changes used in the security proofs of the above two results, i.e. that modified 2-Feistel realizes UC RIC, and that EKE with encryption using HIC realizes UC PAKE. It also contains the security proof of the EKE-KEM protocol, and the proof that the KHAPE protocol of [33] realizes UC aPAKE with IC encryption replaced by HIC. It also illustrates an instantiation of EKE-KEM protocol with Saber KEM [23], and compares the resulting protocol to prior lattice-based PAKEs.

2 Preliminaries

We focus our treatment of the EKE protocol to instantiations that use Key Exchange (KE) with either a single simultaneous flow or 2 flows. Since a 2-flow KE is equivalent to a key encapsulation mechanism (KEM), we will use "KE" to refer to a single-round key exchange, and "KEM" to a KEM *and* to a two-flow key exchange implied by it.

2.1 Single-round Key Exchange (KE) Scheme

A (single-round) KE scheme is a pair of algorithms $\mathsf{KA} = (\mathsf{msg}, \mathsf{key})$, where:

- msg, on input a security parameter κ, generates message M and state x;
- key, on input state x and incoming message M', generates session key K.

The correctness requirement is that if two parties exchange honestly generated messages then they both output the same session key, i.e. if $(x_1, M_1) \leftarrow \mathsf{msg}(1^\kappa)$ and $(x_2, M_2) \leftarrow \mathsf{msg}(1^\kappa)$ then $\mathsf{key}(x_1, M_2) = \mathsf{key}(x_2, M_1)$. The KE security requirement is that a KE transcript hides the session key, but as noted by Bellare et al. [6], the EKE protocol requires an additional property of KE called a *random-message* property, namely that messages output by msg are indistinguishable from values sampled from a uniform distribution over some domain $\mathcal{M}$. (In the security analysis of EKE by [6], the EKE employs an Ideal Cipher on domain $\mathcal{M}$ for password-encryption of KE protocol messages.)

Definition 1. *KE scheme* $(\mathsf{msg}, \mathsf{key})$ *is* secure *if distributions* $\{(M_1, M_2, K)\}$ *and* $\{(M_1, M_2, K^*)\}$ *are computationally indistinguishable, where* $(x_1, M_1) \leftarrow \mathsf{msg}(1^\kappa)$, $(x_2, M_2) \leftarrow \mathsf{msg}(1^\kappa)$, $K \leftarrow \mathsf{key}(x_1, M_2)$, *and* $K^* \xleftarrow{r} \{0,1\}^\kappa$.

Definition 2. *KE scheme* $(\mathsf{msg}, \mathsf{key})$ *has the* random-message *property on domain* $\mathcal{M}$*, indexed by sec. par.* κ*, if the distribution* $\{M \mid (x, M) \leftarrow \mathsf{msg}(1^\kappa)\}$ *is computationally indistinguishable from uniform over set* $\mathcal{M}[\kappa]$.

2.2 Key Encapsulation Mechanism (KEM)

A KEM scheme is a tuple of efficient algorithms $\mathsf{KEM} = (\mathsf{kg}, \mathsf{enc}, \mathsf{dec})$, where:

- kg, on input secpar κ, generates public and private keys pk and sk;
- enc, on input a public key pk, generates ciphertext e and session key K;
- dec, on input a private key sk and a ciphertext e, outputs a session key K.

The correctness requirement is that if $(sk, pk) \leftarrow \mathsf{kg}(1^\kappa)$ and $(e, K) \leftarrow \mathsf{enc}(pk)$ then $\mathsf{dec}(sk, e) = K$. Note that KEM models any 2-flow key exchange scheme, where the public key pk is the initiator's message, and the ciphertext e is the responder's message. We require IND security of KEM, and two additional randomness/anonymity properties: First, public keys must be *uniform* in the sense that their distribution must be indistinguishable from a uniform distribution over some set $\mathcal{PK}$. Secondly, KEM must be anonymous [5], i.e.

ciphertexts must be unlinkable to public keys. Note that these are slightly weaker properties than we asked of KA. Since a key exchange implied by KEM takes 2 flows, the EKE variant using KEM, see Figure 10 in Section 5.1, can use the (randomized) ideal cipher only for the first flow, i.e. the public key, while the second flow, i.e. the KEM ciphertext, can be sent as is, as long as the responder attaches to it a key confirmation message. Consequently, the second message must be unlinkable to the first, but it does not have to be indistinguishable from a random element in a domain of an ideal cipher.

Definition 3. *KEM scheme is* IND secure *if distributions* $\{(pk, e, K)\}$ *and* $\{(pk, e, K^*)\}$ *are computationally indistinguishable, where* $(sk, pk) \xleftarrow{r} \mathsf{kg}(1^\kappa)$, $(e, K) \xleftarrow{r} \mathsf{enc}(pk)$ *and* $K^* \xleftarrow{r} \{0,1\}^\kappa$.

Definition 4. *KEM scheme has* uniform public keys *for domain* $\mathcal{PK}$*, indexed by the security parameter* κ*, if the distribution* $\{pk \,|\, (sk, pk) \xleftarrow{r} \mathsf{kg}(1^\kappa)\}$ *is computationally indistinguishable from uniform over set* $\mathcal{PK}[\kappa]$

Definition 5. *KEM scheme is* anonymous *if distributions* $\{(pk_0, pk_1, e_0)\}$ *and* $\{(pk_0, pk_1, e_1)\}$ *are computationally indistinguishable, where* $(sk_0, pk_0) \xleftarrow{r} \mathsf{kg}(1^\kappa)$, $(sk_1, pk_1) \xleftarrow{r} \mathsf{kg}(1^\kappa)$, $(e_0, K_0) \xleftarrow{r} \mathsf{enc}(pk_0)$, *and* $(e_1, K_1) \xleftarrow{r} \mathsf{enc}(pk_1)$.

Note that the last two properties are trivially achieved by the Diffie-Hellman KEM, where both the public keys and ciphertexts are random group elements. However, both properties are also achieved by several lattice-based KEM's, as discussed in Section 1.

3 Universally Composable Half-Ideal Cipher

We define a new functionality $\mathcal{F}_{\mathsf{HIC}}$ in the UC framework ([15]), called a *(Randomized) Half-Ideal Cipher* (HIC), where the 'half' in the name refers to the fact that only half of the ciphertext is random to the adversary during encryption, as we explain below.

UC HIC is a weakening of the UC Ideal Cipher notion. Intuitively, we allow adversaries to predict or control part of the output of the cipher while the remainder is indistinguishable from random just as in the case of IC. Formally, we can interpret this as allowing the adversary to embed some tuples in the table that the functionality uses - but in a very controlled manner. We define the UC notion of Half-Ideal Cipher via functionality $\mathcal{F}_{\mathsf{HIC}}$ defined in Figure 2.[6]

Notes on $\mathcal{F}_{\mathsf{HIC}}$ interfaces. A half-ideal cipher functionality $\mathcal{F}_{\mathsf{HIC}}$ is parametrized by the (randomized) cipher domain $\mathcal{D} = \mathcal{R} \times \mathcal{G}$, where the first component is the randomness and the second is the plaintext. Figure 2 separates between $\mathcal{F}_{\mathsf{HIC}}$ interfaces Enc and Dec which are used by honest

[6] In Figure 2 we use *pw* to denote keys used in the HIC cipher because we use variables k and K for other keys in the later sections. Moreover, in PAKE and aPAKE applications the role of a HIC key is played by a password.

Notation: Functionality $\mathcal{F}_{\mathsf{HIC}}$ is parametrized by domain $\mathcal{D} = \mathcal{R} \times \mathcal{G}$, and it is indexed by a session identifier sid which is a global constant, hence we omit it from notation. We denote HIC keys as passwords pw to conform to the usage of $\mathcal{F}_{\mathsf{HIC}}$ in PAKE and aPAKE applications, but keys pw are arbitrary bitstrings.

Initialization: For all $pw \in \{0,1\}^*$, initialize THIC_{pw} as an empty table.

Interfaces for Honest Parties P*:*

on query (Enc, pw, M) from party P, for $M \in \mathcal{G}$:

$r \xleftarrow{r} \mathcal{R}$
if $\exists c$ s.t. $((r, M), c) \in \mathsf{THIC}_{pw}$ then return c to P, else do:
  $c \xleftarrow{r} \{\hat{c} \in \mathcal{D} : \nexists m$ s.t. $(m, \hat{c}) \in \mathsf{THIC}_{pw}\}$
  add $((r, M), c)$ to THIC_{pw} and return c to P

on query (Dec, pw, c) from party P, for $c \in \mathcal{D}$:

query $(r, M) \leftarrow \mathcal{F}_{\mathsf{HIC}}.\mathsf{AdvDec}(pw, c)$ and return M to P

Interfaces for Adversary $\mathcal{A}$ *(or corrupt parties):*

on query $(\mathsf{AdvEnc}, pw, (r, M), T)$ from adversary $\mathcal{A}$, for $(r, M) \in \mathcal{D}$ and $T \in \mathcal{G}$:

if $\exists c$ s.t. $((r, M), c) \in \mathsf{THIC}_{pw}$ then return c to $\mathcal{A}$, else do:
  $s \xleftarrow{r} \{\hat{s} \in \mathcal{R} : \nexists \hat{m}$ s.t. $(\hat{m}, (\hat{s}, T)) \in \mathsf{THIC}_{pw}\}$
  set $c \leftarrow (s, T)$, add $((r, M), c)$ to THIC_{pw}, and return c to $\mathcal{A}$

on query (AdvDec, pw, c) from adversary $\mathcal{A}$, for $c \in \mathcal{D}$:

if $\exists m$ s.t. $(m, c) \in \mathsf{THIC}_{pw}$ then return m to $\mathcal{A}$, else do:
  $m \xleftarrow{r} \{\hat{m} \in \mathcal{D} : \nexists \hat{c}$ s.t. $(\hat{m}, \hat{c}) \in \mathsf{THIC}_{pw}\}$
  add (m, c) to THIC_{pw} and return m to $\mathcal{A}$

Fig. 2: Ideal functionality $\mathcal{F}_{\mathsf{HIC}}$ for *(Randomized) Half-Ideal Cipher* on $\mathcal{D} = \mathcal{R} \times \mathcal{G}$

parties, and the adversarial interfaces AdvEnc and AdvDec. Interfaces Enc and Dec model honest-party's usage of HIC, i.e. a real-world implementation of HIC will consists of two algorithms, Enc and Dec, where Enc on input key pw and plaintext $M \in \mathcal{G}$ outputs a ciphertext $c \in \mathcal{D}$ and Dec on input key pw and ciphertext $c \in \mathcal{D}$ outputs a plaintext $M \in \mathcal{G}$. Our target realization of these procedures is a *randomized cipher*, i.e. a family of functions Π_{pw} s.t. for each $pw \in \{0,1\}^*$, Π_{pw} is a permutation on $\mathcal{D}$, and both Π_{pw} and Π_{pw}^{-1} are efficiently evaluable given pw. Given cipher Π, algorithm $\mathsf{Enc}(pw, M)$ picks $r \xleftarrow{r} \mathcal{R}$ and outputs $c \leftarrow \Pi_{pw}(m)$ for $m = (r, M)$, while $\mathsf{Dec}(pw, c)$ computes $m \leftarrow \Pi_{pw}^{-1}(c)$ and output M for $(r, M) = m$.

Functionality walk-through. Functionality $\mathcal{F}_{\mathsf{HIC}}$ reflects honest user's interfaces to randomized encryption: When an honest party P encrypts a message it specifies only $M \in \mathcal{G}$ and delegates the choice of randomness $r \xleftarrow{r} \mathcal{R}$ to the functionality. Similarly, when an honest party decrypts a ciphertext, the functionality discards the randomness r and reveals only M to the application.

This implies that honest parties must use fresh randomness at each encryption and must discard it (or at least not use it) at decryption. By contrast, an adversary $\mathcal{A}$ has stronger interfaces than honest parties (for notational simplicity we assume corrupt parties interact to $\mathcal{F}_{\mathsf{HIC}}$ via $\mathcal{A}$), namely: (1) When $\mathcal{A}$ encrypts it can choose randomness r at will; (2) When $\mathcal{A}$ decrypts it learns the randomness r and does not have to discard it; (3) $\mathcal{A}$ can manipulate the (plaintext, ciphertext) table of each permutation Π_{pw} in the following way: If we denote ciphertexts as $c = (s, T) \in \mathcal{R} \times \mathcal{G}$, the adversary has no control of the s component of the ciphertext at encryption, i.e. it is random in $\mathcal{R}$ (up to the fact that the map has to remain a permutation), but the adversary can freely choose the T component. Items (1) and (2) are consequences of the fact that HIC is a *randomized* cipher, but item (3) is what makes this cipher *Half-Ideal*, because the adversary can control part of the value $c = \mathsf{Enc}(pw, m)$ during encryption, namely its $\mathcal{G}$ component.

The above relaxations of Ideal Cipher (IC) properties are imposed by the modified 2-Feistel construction, which in Section 4 we show realizes this model. However, this relaxation is harmless for many IC applications the following reason: In a typical IC application the benefit of ciphertext randomness is that it (1) hides the plaintext, and (2) it prevents the adversary from creating the same ciphertext as an encryption of two different plaintexts under two different keys. For both purposes randomness in the $s \in \mathcal{R}$ component of the ciphertext suffices as long as $\mathcal{R}$ is large enough to prevent ever encountering collisions.

The adversarial interfaces AdvEnc and AdvDec of $\mathcal{F}_{\mathsf{HIC}}$ reflect the above, and give more powers than the honest party's interfaces Enc and Dec. In encryption query AdvEnc, the adversary is allowed to pick its own randomness r *and* the $T \in \mathcal{G}$ part of the resulting ciphertext, while its s part is chosen at random in $\mathcal{R}$. In decryption AdvDec, the adversary can decrypt any ciphertext $c = (s, T)$ and it learns the full plaintext $m = (r, M)$, but $\mathcal{F}_{\mathsf{HIC}}$ chooses the whole plaintext m at random. (This is another motivation for the monicker 'half-ideal': $\mathcal{F}_{\mathsf{HIC}}$ lets the adversary have some control over ciphertexts in encryption but it does not let the adversary have any control over plaintexts in decryption.)

Our goal when designing $\mathcal{F}_{\mathsf{HIC}}$ was to keep all IC properties which are useful in applications while allowing for efficient concrete instantiation of $\mathcal{F}_{\mathsf{HIC}}$ for a group domain $\mathcal{G}$. Most importantly, ciphertext collisions in encryption can occur only with negligible probability, which is crucial in our HIC applications: An adversarial ciphertext c commits the adversary to a single key pw on which the adversary could have computed c as an encryption of some message of its choice. Secondly, just as with an ideal cipher, the adversary cannot learn any information on encrypted plaintexts except via decryption with correct decryption key.

4 Half-Ideal Cipher Construction: Modified 2-Feistel

We modify the two-round Feistel construction of the Programmable Once Public Functions (POPF) of McQuoid et al. [43] by replacing the xor operation in the second round by an application of an ideal block cipher BC on bitstrings, with

keys and plaintext block both of size 2κ where κ is the security parameter. We call this construction a *modified 2-Feistel*, denoted $\mathsf{m2F}$. This construction takes (1) an ideal cipher BC on bitstrings, i.e. an ideal cipher whose domain is $\{0,1\}^n$ and key space is $\{0,1\}^\mu$, (2) a random oracle hash H' with range $\{0,1\}^\mu$, and (3) a random oracle hash H whose range is an arbitrary *group* $\mathbb{G}$, and creates a *(Randomized) Half-Ideal Cipher* (HIC) over domain $\mathcal{D} = \mathcal{R} \times \mathbb{G}$ where $\mathcal{R} = \{0,1\}^n$. In essence, we combine a random oracle hash onto a group and a bitwise ideal cipher to create a half-ideal cipher over a group. The exact security analysis of the $\mathsf{m2F}$ construction shows that μ and n can both be set to 2κ for this construction to realize UC HIC.

For each key pw, function $\mathsf{m2F}_{pw}$ is pictorially shown in Figure 1. Here we define it by the algorithms which compute $\mathsf{m2F}_{pw}$ and $\mathsf{m2F}_{pw}^{-1}$. (Throughout the paper we denote group $\mathbb{G}$ operation as a multiplication, but this is purely a notational choice, and the construction applies to additive groups as well.)

$$\mathsf{m2F}_{pw} \ : \ \{0,1\}^n \times \mathbb{G} \ \ \rightarrow \ \ \{0,1\}^n \times \mathbb{G} \tag{1}$$

where:

$\mathsf{m2F}_{pw}(r, M)$:

1. $T \leftarrow M/\mathsf{H}(pw, r)$
2. $k \leftarrow \mathsf{H}'(pw, T)$
3. $s \leftarrow \mathsf{BC.Enc}(k, r)$
4. Output (s, T)

$\mathsf{m2F}_{pw}^{-1}(s, T)$:

1. $k \leftarrow \mathsf{H}'(pw, T)$
2. $r \leftarrow \mathsf{BC.Dec}(k, s)$
3. $M \leftarrow \mathsf{H}(pw, r) \cdot T$
4. Output (r, M)

The following theorem captures the security of the $\mathsf{m2F}$ construction:

Theorem 1. *Construction* $\mathsf{m2F}$ *realizes functionality* $\mathcal{F}_{\mathsf{HIC}}$ *in the domain* $\mathcal{R} \times \mathbb{G}$ *for* $\mathcal{R} = \{0,1\}^n$ *if* $\mathsf{H} : \{0,1\}^* \times \{0,1\}^n \rightarrow \mathbb{G}$, $\mathsf{H}' : \{0,1\}^* \times \mathbb{G} \rightarrow \{0,1\}^\mu$ *are random oracles,* $\mathsf{BC} : \{0,1\}^\mu \times \{0,1\}^n \rightarrow \{0,1\}^n$ *is an ideal cipher, and* μ *and* n *are both* $\Omega(\kappa)$.

Proof. The proof for Theorem 1 must exhibit a simulator algorithm SIM, which plays a role of an ideal-world adversary interacting with functionality $\mathcal{F}_{\mathsf{HIC}}$, and then show that no efficient environment $\mathcal{Z}$ can distinguish, except for negligible probability, between (1) a *real-world game*, i.e. an interaction with (1a) honest parties who execute $\mathcal{Z}$'s encryption and decryption queries using Enc and Dec implemented with circuit $\mathsf{m2F}$ (see Section 3), and (1b) RO/IC oracles H, H', BC, BC^{-1}, and (2) an *ideal-world game*, i.e. an interaction with (2a) parties P who execute $\mathcal{Z}$'s encryption and decryption using interfaces $\mathsf{Enc}, \mathsf{Dec}$ of $\mathcal{F}_{\mathsf{HIC}}$, and (2b) simulator SIM, who services $\mathcal{Z}$'s calls to H, H', BC, BC^{-1} using interfaces AdvEnc and AdvDec of $\mathcal{F}_{\mathsf{HIC}}$.

We start by describing the simulator algorithm SIM, shown in Figure 3. Note that SIM interacts with an adversarial environment algorithm $\mathcal{Z}$ by servicing $\mathcal{Z}$'s queries to the RO and IC oracles $\mathsf{H}, \mathsf{H}', \mathsf{BC}, \mathsf{BC}^{-1}$. Intuitively, SIM populates input, output tables for these functions, $\mathsf{TH}, \mathsf{TH}'$ and TBC, in the same way as these idealized oracles would, except when SIM detects a possible encryption or

Initialization

Let TH be a set of tuples in $\{0,1\}^* \times \{0,1\}^n \times \mathbb{G}$,
TH' be a set of tuples in $\{0,1\}^* \times \mathbb{G} \times \{0,1\}^\mu$,
and TBC be a set of triples in $\{0,1\}^\mu \times \{0,1\}^n \times \{0,1\}^n$.

on adversary's query $\mathsf{H}(pw, r)$

if $\nexists h$ s.t. $(pw, r, h) \in \mathsf{TH}$:
  $h \xleftarrow{r} \mathbb{G}$
  add (pw, r, h) to TH
return h

on adversary's query $\mathsf{H}'(pw, T)$

if $\nexists k$ s.t. $(pw, T, k) \in \mathsf{TH}'$:
  $k \xleftarrow{r} \{0,1\}^\mu$
  if $\exists (\hat{pw}, \hat{T})$ s.t. $(\hat{pw}, \hat{T}, k) \in \mathsf{TH}'$ then abort (col.abort)
  if $\exists (\hat{r}, \hat{s})$ s.t. $(k, \hat{r}, \hat{s}) \in \mathsf{TBC}$ then abort (bckey.abort)
  add (pw, T, k) to TH'
return k

on adversary's query $\mathsf{BC.Enc}(k, r)$

if $\nexists s$ s.t. $(k, r, s) \in \mathsf{TBC}$:
  if $k = \mathsf{TH}'(pw, T)$[a]:
    $M \leftarrow \mathsf{H}(pw, r) \cdot T$
    $(s, \hat{T}) \leftarrow \mathcal{F}_{\mathsf{HIC}}.\mathsf{AdvEnc}(pw, (r, M), T)$
    if $\hat{T} \neq T$ then abort (advenc.abort)
  else:
    $s \xleftarrow{r} \{s \in \{0,1\}^n : \nexists \hat{r}$ s.t. $(k, \hat{r}, s) \in \mathsf{TBC}\}$
  add (k, r, s) to TBC
return s

on adversary's query $\mathsf{BC.Dec}(k, s)$

if $\nexists r$ s.t. $(k, r, s) \in \mathsf{TBC}$:
  if $k = \mathsf{TH}'(pw, T)$:
    $(r, M) \leftarrow \mathcal{F}_{\mathsf{HIC}}.\mathsf{AdvDec}(pw, (s, T))$
    if $\exists \hat{s}$ s.t. $(k, r, \hat{s}) \in \mathsf{TBC}$ then abort (advdec.abort)
    if $\exists h$ s.t. $(pw, r, h) \in \mathsf{TH}$ then abort (rcol.abort)
    add $(pw, r, M \cdot T^{-1})$ to TH
  else:
    $r \xleftarrow{r} \{r \in \{0,1\}^n : \nexists \hat{s}$ s.t. $(k, r, \hat{s}) \in \mathsf{TBC}\}$
  add (k, r, s) to TBC
return r

[a] If it exists, we denote by $\mathsf{TH}'(pw, T)$ the (unique) k s.t. $(pw, T, k) \in \mathsf{TH}'$

Fig. 3: Simulator SIM for the proof of Theorem 1

decryption computation of the modified 2-Feistel circuit. In case SIM decides that these queries form either computation of $\mathsf{m2F}$ or $\mathsf{m2F}^{-1}$ on new input, SIM detects that input, invokes the adversarial interfaces AdvEnc or AdvDec of $\mathcal{F}_{\mathsf{HIC}}$ to find the corresponding output, and it embeds proper values into these tables to emulate the circuit leading to the computation of this output. The detection of $\mathsf{m2F}$ and $\mathsf{m2F}^{-1}$ evaluation is relatively straightforward: First, SIM treats every $\mathsf{BC.Dec}$ query (k, s) as a possible $\mathsf{m2F}^{-1}$ evaluation on key pw and ciphertext $c = (s, T)$ for T s.t. $k = \mathsf{H}'(pw, T)$. If it is, SIM queries $\mathcal{F}_{\mathsf{HIC}}.\mathsf{AdvDec}$ on (pw, c) to get $m = (r, M)$. Since this is a random sample from the HIC domain, with overwhelming probability H was not queried on r so SIM can set $\mathsf{H}(pw, r)$ to M/T. Second, SIM treats every $\mathsf{BC.Enc}$ query (k, r) as possible $\mathsf{m2F}$ evaluation on (r, M) s.t. $M = \mathsf{H}(pw, r) \cdot T$ for T s.t. $k = \mathsf{H}'(pw, T)$. However, here is where the difference between IC and HIC shows up: The $\mathcal{F}_{\mathsf{HIC}}.\mathsf{AdvEnc}$ query fixes the encryption of $m = (r, M)$ to $c = (s, T)$, and whereas s can be random (and SIM can set $\mathsf{BC.Enc}(k, r) := s$ for any $c = (s, T)$ returned by $\mathcal{F}_{\mathsf{HIC}}.\mathsf{AdvEnc}$ as encryption of m under key pw), value T was fixed by H' output k (except for the negligible probability of finding collisions in H'). This is why our $\mathcal{F}_{\mathsf{HIC}}$ model

must allow the simulator, i.e. the ideal-world adversary, to fix the T part of the ciphertext in the adversarial encryption query AdvEnc.

Proof Overview. The proof must show that for any environment $\mathcal{Z}$, its view of the real-world game defined by algorithms $\mathsf{Enc}, \mathsf{Dec}$ which use the randomized cipher $\mathsf{m2F}$, and the ideal-world game defined by functionality $\mathcal{F}_{\mathsf{HIC}}$ and simulator SIM of Figure 3. The proof starts from the ideal-world view, which we denote as Game 0, and via a sequence of games, each of which we show is indistinguishable from the next, it reaches the real-world view, which we denote as Game 9. For space-constraint reasons we include the details of the game changes and reductions to the full version [30], but we show the code of all successive games in Figures 4, 5, and 6. Figure 4 describes the ideal-world Game 0 and its mild modification Game 1. All these games, starting from Game 0 in Figure 4, interact with an adversarial environment $\mathcal{Z}$, and each game provides two types of interfaces corresponding two types of $\mathcal{Z}$'s queries: (a) the honest party's interfaces $\mathsf{Enc}, \mathsf{Dec}$, which $\mathcal{Z}$ can query via any honest party, and (b) RO/IC oracles $\mathsf{H}, \mathsf{H}', \mathsf{BC}, \mathsf{BC}^{-1}$, which $\mathcal{Z}$ can query via its "real-world adversary" interface. Figure 4 defines two sub-procedures, $\mathcal{F}_{\mathsf{HIC}}.\mathsf{AdvEnc}$ and $\mathcal{F}_{\mathsf{HIC}}.\mathsf{AdvDec}$, whose code matches exactly the corresponding interfaces of $\mathcal{F}_{\mathsf{HIC}}$. These subprocedures are used internally by Game 0: They are invoked by the code that services $\mathcal{Z}$'s queries $\mathsf{BC.Enc}$ and $\mathsf{BC.Dec}$, because Game 0 follows SIM's code on these queries, and AdvDec is also invoked by Dec, because this is how $\mathcal{F}_{\mathsf{HIC}}$ implements Dec.

Figures 5 and 6 describe the modifications created by all subsequent games, except for the last one, the real-world game denoted Game 9, which is very similar to Game 8, which is the last game shown in Figure 6. By the arguments for indistinguishability of successive games shown in the full version [30], the total distinguishing advantage of environment $\mathcal{Z}$ between the real-world and the ideal-world interaction is upper-bounded by the following expression, which sums up the bounds given by equations shown in the proof, see [30]:

$$|P_0 - P_9| \leq q^2 \left(\frac{10}{2^n} + \frac{4}{2^n \cdot |\mathbb{G}|} + \frac{6}{2^\mu} \right) \leq q^2 \left(\frac{14}{2^n} + \frac{6}{2^\mu} \right)$$

Since this quantity is negligible, this implies Theorem 1 □

Notes on Exact Security. By the above equation, the distinguishability advantage implies by our proof can be upper-bounded as $O(q^2/2^n) + O(q^2/2^\mu)$. We assert that both of these factors are unavoidable for our $\mathsf{m2F}$ construction. First, while in the $\mathcal{F}_{\mathsf{HIC}}$ functionality we allow the T component of two AdvEnc adversarial calls to be completely independent, this is not the case in our modified two-round Feistel encryption: reuse of a (pw, r) pair implies relations between the T component of different encryption calls that are not seen in $\mathcal{F}_{\mathsf{HIC}}$. Hence we must avoid r collisions in Enc calls, irrespective of how our proof is structured, and asymptotically this gives a $q^2/2^n$ factor in the distinguishing advantage.

Initialization

Let TH be a set of tuples in $\{0,1\}^* \times \{0,1\}^n \times \mathbb{G}$,
TH' be a set of tuples in $\{0,1\}^* \times \mathbb{G} \times \{0,1\}^\mu$,
and TBC be a set of triples in $\{0,1\}^\mu \times \{0,1\}^n \times \{0,1\}^n$.

For each $pw \in \{0,1\}^*$, initialize empty sets THIC_{pw} and usedR_{pw}.

define $\mathcal{F}_{\mathsf{HIC}}.\mathsf{AdvEnc}(pw,(r,M),T)$:

if $\nexists c$ s.t. $((r,M),c) \in \mathsf{THIC}_{pw}$:
  $s \xleftarrow{r} \{\hat{s} \in \{0,1\}^n : (*,(\hat{s},T)) \notin \mathsf{THIC}_{pw}\}$
  $c \leftarrow (s,T)$
  add $((r,M),c)$ to THIC_{pw}
return c

define $\mathcal{F}_{\mathsf{HIC}}.\mathsf{AdvDec}(pw,(s,T))$:

if $\nexists (r,M)$ s.t. $((r,M),(s,T)) \in \mathsf{THIC}_{pw}$:
  $(r,M) \xleftarrow{r} \mathcal{D}$
  if $\exists \hat{c}$ s.t. $((r,M),\hat{c}) \in \mathsf{THIC}_{pw}$ then abort
  abort if $r \in \mathsf{usedR}_{pw}$ else add r with tag $\mathsf{m2F}$
  add $((r,M),(s,T))$ to THIC_{pw}
return M

on query $\mathsf{Enc}(pw,M)$:

$r \xleftarrow{r} \{0,1\}^n$
abort if $r \in \mathsf{usedR}_{pw}$, else add r with tag $\mathsf{m2F}$
if $\nexists c$ s.t. $((r,M),c) \in \mathsf{THIC}_{pw}$:
  $c \xleftarrow{r} \{\hat{c} : \nexists \hat{m}$ s.t. $(\hat{m},\hat{c}) \in \mathsf{THIC}_{pw}\}$
  add $((r,M),c)$ to THIC_{pw}
return c

on query $\mathsf{Dec}(pw,c)$:

$(r,M) \leftarrow \mathcal{F}_{\mathsf{HIC}}.\mathsf{AdvDec}(pw,c)$
return M

on query $\mathsf{H}(pw,r)$

abort if $r \in \mathsf{usedR}_{pw}$ tagged $\mathsf{m2F}$, else add r
if $\nexists h$ s.t. $(pw,r,h) \in \mathsf{TH}$:
  $h \xleftarrow{r} \mathbb{G}$
  add (pw,r,h) to TH
return h

on query $\mathsf{H}'(pw,T)$

if $\nexists k$ s.t. $(pw,T,k) \in \mathsf{TH}'$:
  $k \xleftarrow{r} \{0,1\}^\mu$
  if $\exists\ (\hat{pw},\hat{T})$ s.t. $(\hat{pw},\hat{T},k) \in \mathsf{TH}'$ then abort (col.abort)
  if $\exists\ (\hat{r},\hat{s})$ s.t. $(k,\hat{r},\hat{s}) \in \mathsf{TBC}$ then abort (bckey.abort)
  add (pw,T,k) to TH'
return k

on query $\mathsf{BC.Enc}(k,r)$

if $k = \mathsf{TH}'(pw,T)$:
  if $r \in \mathsf{usedR}_{pw}$ is tagged $\mathsf{m2F}$ then abort
  else add r to usedR_{pw}
if $\nexists s$ s.t. $(k,r,s) \in \mathsf{TBC}$:
  if $k = \mathsf{TH}'(pw,T)$:
    $M \leftarrow \mathsf{H}(pw,r) \cdot T$
    $(s,\hat{T}) \leftarrow \mathcal{F}_{\mathsf{HIC}}.\mathsf{AdvEnc}(pw,(r,M),T)$
    if $\hat{T} \neq T$ then abort (advenc.abort)
  else:
    $s \xleftarrow{r} \{s \in \{0,1\}^n : \nexists \hat{r}$ s.t. $(k,\hat{r},s) \in \mathsf{TBC}\}$
  add (k,r,s) to TBC
return s

on query $\mathsf{BC.Dec}(k,s)$

if $\nexists r$ s.t. $(k,r,s) \in \mathsf{TBC}$:
  if $k = \mathsf{TH}'(pw,T)$:
    $(r,M) \leftarrow \mathcal{F}_{\mathsf{HIC}}.\mathsf{AdvDec}(pw,(s,T))$
    if $\exists \hat{s}$ s.t. $(k,r,\hat{s}) \in \mathsf{TBC}$ then abort (advdec.abort)
    if $\exists\ h$ s.t. $(pw,r,h) \in \mathsf{TH}$ then abort (rcol.abort)
    add $(pw,r,M \cdot T^{-1})$ to TH
  else:
    $r \xleftarrow{r} \{r \in \{0,1\}^n : \nexists \hat{s}$ s.t. $(k,r,\hat{s}) \in \mathsf{TBC}\}$
  add (k,r,s) to TBC
if $k = \mathsf{TH}'(pw,T)$:
  remove tag $\mathsf{m2F}$ from record $r \in \mathsf{usedR}_{pw}$
return r

Fig. 4: The ideal-world Game 0, and its modification Game 1 (text in gray)

Game 2: replacing decryption by circuit

on query $\mathsf{m2F.Dec}(pw,(s,T))$:
$k \leftarrow \mathsf{H}'(pw,T)$
$r \leftarrow \mathsf{BC.Dec}(k,s)$
$M \leftarrow \mathsf{H}(pw,r)\cdot T$
if m2F.Dec query was fresh, add tag m2F to $r \in \mathsf{usedR}_{pw}$
return M

Game 3: Enc calls AdvDec

on query $\mathsf{m2F.Enc}(pw,M)$:
$r \xleftarrow{r} \{0,1\}^n$
if $r \in \mathsf{usedR}_{pw}$ abort, else add r to it with tag m2F
if $\nexists c$ s.t. $((r,M),c) \in \mathsf{THIC}_{pw}$:
  $T \xleftarrow{r} \mathbb{G}$
  $c \leftarrow \mathcal{F}_{\mathsf{HIC}}.\mathsf{AdvEnc}(pw,(r,M),T)$
return c

Game 4: replacing encryption by circuit

on query $\mathsf{m2F.Enc}(pw,M)$:
$r \xleftarrow{r} \{0,1\}^n$
if $r \in \mathsf{usedR}_{pw}$ abort
$T \leftarrow M/\mathsf{H}(pw,r)$
$k \leftarrow \mathsf{H}'(pw,T)$
$s \leftarrow \mathsf{BC.Enc}(k,r)$
assign tag m2F to r in the set usedR_{pw}
return (s,T)

Game 5: H is a random oracle

$\mathcal{F}_{\mathsf{HIC}}.\mathsf{AdvDec}$ not used anymore

on query $\mathsf{BC.Dec}(k,s)$:

if $\nexists\ r$ s.t. $(k,r,s) \in \mathsf{TBC}$:
  if $k = \mathsf{TH}'(pw,T)$:
    $r \xleftarrow{r} \{0,1\}^n$
    if $r \in \mathsf{usedR}_{pw}$ abort, else add r to it
    $h \leftarrow \mathsf{H}(pw,r)$
    $M \leftarrow h \cdot T$
    if $\exists \hat{c}$ s.t. $((r,M),\hat{c}) \in \mathsf{THIC}_{pw}$ then abort
    add $((r,M),(s,T))$ to THIC_{pw}
  else:
    $r \xleftarrow{r} \{r \in \{0,1\}^n : \nexists \hat{s}$ s.t. $(k,r,\hat{s}) \in \mathsf{TBC}$
  add (k,r,s) to TBC
remove tag m2F from record $r \in \mathsf{usedR}_{pw}$ if $k = \mathsf{TH}'(pw,T)$
return r

Game 6: simplifying parameters

define $\mathcal{F}_{\mathsf{HIC}}.\mathsf{AdvEnc}(pw,r,T)$:
if $\nexists s$ s.t. $(r,(s,T)) \in \mathsf{THIC}_{pw}$:
  $s \xleftarrow{r} \{\hat{s} \in \{0,1\}^n : \nexists \hat{r}$ s.t. $(\hat{r},(\hat{s},T)) \in \mathsf{THIC}_{pw}\}$
  add $(r,(s,T))$ to THIC_{pw}
return s

on query $\mathsf{BC.Dec}(k,s)$:
if $\nexists r$ s.t. $(k,r,s) \in \mathsf{TBC}$:
  if $k = \mathsf{TH}'(pw,T)$:
    $r \xleftarrow{r} \{0,1\}^n$
    if $r \in \mathsf{usedR}_{pw}$ abort, else add r to it
    query $\mathsf{H}(pw,r)$ and discard the output
    if $\exists \hat{c}$ s.t. $(r,\hat{c}) \in \mathsf{THIC}_{pw}$ then abort
    add $(r,(s,T))$ to THIC_{pw}
  else:
    $r \xleftarrow{r} \{r \in \{0,1\}^n : \nexists \hat{s}$ s.t. $(k,r,\hat{s}) \in \mathsf{TBC}$
  add (k,r,s) to TBC
remove tag m2F from record $r \in \mathsf{usedR}_{pw}$ if $k = \mathsf{TH}'(pw,T)$
return r

on query $\mathsf{BC.Enc}(k,r)$:
if $k = \mathsf{TH}'(pw,T)$:
  if $r \in \mathsf{usedR}_{pw}$ is tagged m2F then abort
    else add r to usedR_{pw}
if $\nexists s$ s.t. $(k,r,s) \in \mathsf{TBC}$:
  if $k = \mathsf{TH}'(pw,T)$:
    query $\mathsf{H}(pw,r)$ and discard the output
    $s \leftarrow \mathcal{F}_{\mathsf{HIC}}.\mathsf{AdvEnc}(pw,r,T)$
  else:
    $s \xleftarrow{r} \{s \in \{0,1\}^n : \nexists \hat{r}$ s.t. $(k,\hat{r},s) \in \mathsf{TBC}\}$
  add (k,r,s) to TBC
return s

Game 7: using k

Initialization: $\forall\, k$ initialize empty THIC_k

define $\mathcal{F}_{\mathsf{HIC}}.\mathsf{AdvEnc}(k,r)$:
if $\nexists s$ s.t. $(r,s) \in \mathsf{THIC}_k$:
  $s \xleftarrow{r} \{\hat{s} \in \{0,1\}^n : \nexists \hat{r}$ s.t. $(\hat{r},\hat{s}) \in \mathsf{THIC}_k\}$
  add (r,s) to THIC_k
return s

on query $\mathsf{BC.Dec}(k,s)$:
if $\nexists r$ s.t. $(k,r,s) \in \mathsf{TBC}$:
  if $k = \mathsf{TH}'(pw,T)$:
    $r \xleftarrow{r} \{0,1\}^n$
    if $r \in \mathsf{usedR}_{pw}$ abort, else add r to it
    if $\exists \hat{s}$ s.t. $(r,\hat{s}) \in \mathsf{THIC}_k$ then abort
    add (r,s) to THIC_k
  else:
    $r \xleftarrow{r} \{r \in \{0,1\}^n : \nexists \hat{s}$ s.t. $(k,r,\hat{s}) \in \mathsf{TBC}$
  add (k,r,s) to TBC
remove tag m2F from $r \in \mathsf{usedR}_{pw}$ if $k = \mathsf{TH}'(pw,T)$
return r

on query $\mathsf{BC.Enc}(k,r)$:
if $k = \mathsf{TH}'(pw,T)$:
  if $r \in \mathsf{usedR}_{pw}$ is tagged m2F then abort
    else add r to usedR_{pw}
if $\nexists s$ s.t. $(k,r,s) \in \mathsf{TBC}$:
  if $k = \mathsf{TH}'(pw,T)$:
    $s \leftarrow \mathcal{F}_{\mathsf{HIC}}.\mathsf{AdvEnc}(k,r)$
  else:
    $s \xleftarrow{r} \{s \in \{0,1\}^n : \nexists \hat{r}$ s.t. $(k,\hat{r},s) \in \mathsf{TBC}\}$
  add (k,r,s) to TBC
return s

Fig. 5: Game-changes (part 1) in the proof of Theorem 1

Game 8: THIC is redundant

Initialization: Drop THIC usage.
$\mathcal{F}_{\mathsf{HIC}}.\mathsf{AdvEnc}$ not used anymore

on query $\mathsf{BC.Enc}(k, r)$:
if $k = \mathsf{TH}'(pw, T)$:
  if $r \in \mathsf{usedR}_{pw}$ is tagged $\mathsf{m2F}$, abort, else add $r \in \mathsf{usedR}_{pw}$
if $\nexists s$ s.t. $(k, r, s) \in \mathsf{TBC}$:
  $s \xleftarrow{r} \{s \in \{0,1\}^n : \nexists \hat{r}$ s.t. $(k, \hat{r}, s) \in \mathsf{TBC}\}$
  add (k, r, s) to TBC
return s

on query $\mathsf{BC.Dec}(k, s)$:
if $\nexists r$ s.t. $(k, r, s) \in \mathsf{TBC}$:
  if $\exists (pw, T)$ s.t. $(pw, T, k) \in \mathsf{TH}'$:
    $r \xleftarrow{r} \{0,1\}^n$
    if $r \in \mathsf{usedR}_{pw}$ abort, else add r to it
  else:
    $r \xleftarrow{r} \{r \in \{0,1\}^n : \nexists \hat{s}$ s.t. $(k, r, \hat{s}) \in \mathsf{TBC}\}$
  add (k, r, s) to TBC
remove tag $\mathsf{m2F}$ from record $r \in \mathsf{usedR}_{pw}$ if $k = \mathsf{TH}'(pw, T)$
return r

Fig. 6: Game-changes (part 2) in the proof of Theorem 1

Secondly, we need to avoid H' collisions. Indeed, if $\mathsf{H}'(pw, T) = \mathsf{H}'(\hat{pw}, \hat{T})$ then $\mathsf{m2F}$'s decryptions using (pw, T) and $(\hat{pw}, \hat{T})$ create the same $s \mapsto r$ map, which would be in stark contrast to our functionality's ideal-cipher like decryption behavior. We conclude that the $q^2/2^\mu$ term also can't be avoided. Notice that these two terms dominate the probability of the environment distinguishing $\mathsf{m2F}$ from our functionality $\mathcal{F}_{\mathsf{HIC}}$. In particular, they do not involve $|\mathbb{G}|$, i.e., the size of the message space of our $\mathcal{F}_{\mathsf{HIC}}$.

5 Encrypted Key Exchange with Half-Ideal Cipher

We show that the Encrypted Key Exchange (EKE) protocol of Bellovin and Meritt [8] is a universally composable PAKE if the password encryption is implemented with a (Randomized) Half-Ideal Cipher on the domain of messages output by the key exchange scheme, provided that the key exchange scheme has the random-message property (see Section 2). As discussed in the introduction, the same statement was argued by Rosulek et al. [43] with regards to password-encryption implemented using a Programmable Once Public Function (POPF) notion defined therein, which can also be thought of as a weak form of ideal cipher. However, since as we explain in the introduction, the POPF notion is unlikely to suffice in an EKE application, so we need to verify that the notion of UC (Randomized) Half-Ideal Cipher *does* suffice in such application.

In Figure 7 we show the Encrypted Key Exchange protocol EKE, specialized to use a Half-Ideal Cipher for the password-encryption of the message flows of the underlying Key Agreement scheme KA. In Figure 7 we assume that KA is a *single-round* scheme. In Section 5.1 we extend this to the case of two-flow KA, i.e. to EKE protocol instantiated with a KEM scheme. We note that these two treatments are incomparable because in the case of single-flow KA we start from a more restricted KA scheme and we argue security of a single-flow version of EKE, whereas in the case of two-flow KA, i.e. if $\mathsf{KA} = \mathsf{KEM}$, we start from a more general KA scheme but we argue security of a two-flow version of EKE.

The EKE instantiation shown in Figure 7 assumes that the Half-Ideal Cipher HIC works on domain $\mathcal{D} = \mathcal{R} \times \mathcal{M}$ where $\mathcal{M}$ is the message domain of the scheme KA. The "randomness" set $\mathcal{R}$ is arbitrary, but its size influences the security bound we show for such EKE instantiations. In particular we require that $\log(|\mathcal{R}|) \geq 2\kappa$. If HIC is instantiated with the modified 2-Feistel construction m2F of Section 4, one can set $\mathcal{R} = \{0,1\}^{2\kappa}$, and this instantiation of EKE will send messages whose sizes match those of the underlying KA scheme extended by 2κ bits of randomness due to the Half-Ideal Cipher encryption.

In Figure 7 for presentation clarity we assume that party identifiers $\mathsf{P}_0, \mathsf{P}_1$ are lexicographically ordered. The full protocol will use two helper functions order and bit, defined as $\mathsf{order}(\mathsf{sid}, \mathsf{P}, \mathsf{CP}) = (\mathsf{sid}, \mathsf{P}, \mathsf{CP})$ and $\mathsf{bit}(\mathsf{P}, \mathsf{CP}) = 0$ if $\mathsf{P} <_{lex} \mathsf{CP}$, and $\mathsf{order}(\mathsf{sid}, \mathsf{P}, \mathsf{CP}) = (\mathsf{sid}, \mathsf{CP}, \mathsf{P})$ and $\mathsf{bit}(\mathsf{P}, \mathsf{CP}) = 1$ if $\mathsf{CP} <_{lex} \mathsf{P}$ [7]. Party P on input $(\mathsf{NewSession}, \mathsf{sid}, \mathsf{P}, \mathsf{CP}, pw)$ will then set $\mathsf{fullsid} \leftarrow \mathsf{order}(\mathsf{sid}, \mathsf{P}, \mathsf{CP})$ and $b \leftarrow \mathsf{bit}(\mathsf{P}, \mathsf{CP})$ and it will use HIC.Enc on key $\hat{pw}_b = (\mathsf{fullsid}, b, pw)$ to encrypt its outgoing message, and it will use HIC.Dec on key $\hat{pw}_{\neg b} = (\mathsf{fullsid}, \neg b, pw)$ to decrypt its incoming message.

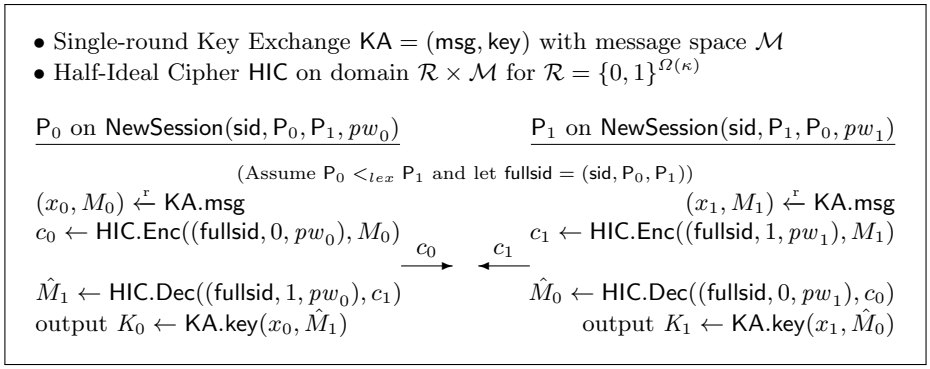

Fig. 7: EKE: Encrypted Key Exchange with Half-Ideal Cipher

In Theorem 2 below we show that protocol EKE realizes the (multi-session version of) the PAKE functionality of Canetti et al. [16], denoted $\mathcal{F}_{\mathsf{pwKE}}$ (e.g., see [30]). The reason we target the multi-session version of PAKE functionality directly, rather than targeting its single-session version and then resorting to Canetti's composition theorem [15] to imply the security of an arbitrary (and concurrent) number of EKE instances, is that for the latter to work we would need the underlying UC HIC to be instantiated separately for each EKE session identifier sid. Our UC HIC notion of Section 3 is a "global" functionality, i.e. it does not natively support separate instances indexed by session identifiers. The modified 2-Feistel construction *could* support such independent instances of HIC by prepending sid to the inputs of all its building block functions $\mathsf{H}, \mathsf{H}', \mathsf{BC}$, where in the last case value sid would have to be prepended to the key of the (ideal)

[7] We assume that no honest P ever executes $(\mathsf{NewSession}, \mathsf{sid}, \mathsf{P}, \mathsf{CP}, \cdot)$ for $\mathsf{CP} = \mathsf{P}$.

block-cipher BC. However, this implies longer inputs for each of these blocks, which is especially problematic in case of the block cipher, so it is preferable not to rely on it and show security for a protocol variant where each EKE instance accesses a single HIC functionality, and hence can be implemented with the same instantiation of the modified 2-Feistel HIC construction.

Theorem 2. *If* KA *is a secure key-exchange scheme with the random-message property on domain* $\mathcal{M}$ *and* HIC *is a UC Half-Ideal Cipher over domain* $\mathcal{R} \times \mathcal{M}$*, then protocol* EKE*, Figure 7, realizes the UC PAKE functionality* $\mathcal{F}_{\mathsf{pwKE}}$*.*

Proof. Let $\mathcal{Z}$ be an arbitrary efficient environment. In the rest of the proof we will assume that the real-world adversary $\mathcal{A}$ is an interface of $\mathcal{Z}$. In Figure 8 we show the construction of a simulator algorithm SIM, which together with functionality $\mathcal{F}_{\mathsf{pwKE}}$ defines the ideal-world view of $\mathcal{Z}$. As is standard, the role of SIM is to emulate actions of honest parties executing protocol EKE given the information revealed by functionality $\mathcal{F}_{\mathsf{pwKE}}$, and to convert the actions of the real-world adversary into queries to $\mathcal{F}_{\mathsf{pwKE}}$. (In Figure 8 we use $\mathsf{P}^{\mathsf{sid}}$ to denote P's session indexed by sid which is emulated by SIM.) The proof then consists of a sequence of games, shown in Figure 9, starting from the real-world game, Game 0, where $\mathcal{Z}$ interacts with the honest parties running protocol EKE, and ending with the ideal-world game, Game 7, where $\mathcal{Z}$ interacts via dummy honest parties with functionality $\mathcal{F}_{\mathsf{pwKE}}$ which in turn interacts with simulator SIM. (This last game is not shown in Figure 9 because its code can be derived from the code of simulator SIM, Figure 8, and functionality $\mathcal{F}_{\mathsf{pwKE}}$, e.g., see [30].) We note that in each game in Figure 9 we write $\boxed{\text{output [...]}}$ for output of queries that service $\mathcal{Z}$'s interaction with EKEinstances, and we write "return [...]" for output of queries that service $\mathcal{Z}$'s interaction with $\mathcal{F}_{\mathsf{HIC}}$.

At each step we prove that the two consecutive games are indistinguishable, which implies the claim by transitivity of computational indistinguishability. Note that we argue security of EKE in the $\mathcal{F}_{\mathsf{HIC}}$-hybrid model. Specifically, algorithm SIM emulates a "global" $\mathcal{F}_{\mathsf{HIC}}$ functionality which services any number of EKE protocol instances. Note that $\mathcal{Z}$ or $\mathcal{A}$ can call $\mathcal{F}_{\mathsf{HIC}}$ on keys which correspond to all strings $\hat{pw} = (\mathsf{fullsid}, b, pw)$ including for fullsid corresponding to sessions which were not (yet) started by $\mathcal{Z}$. Indeed, algorithm SIM treats queries pertaining to any key $\hat{pw}$ equally, and embeds random ciphertext c in response to Enc queries, random partial ciphertext s in response to AdvEnc queries, and random KA message M in response to AdvDec and Dec queries, saving the corresponding KA local state in $(\mathsf{backdoor}, \ldots)$ records. Since Dec is a wrapper over AdvDec we assume that the adversary uses only interface AdvDec, and we implement the EKE code of $\mathsf{P}^{\mathsf{sid}}$ using AdvDec as well.

The intuition for the simulation is that it sends an outgoing EKE message on behalf of $\mathsf{P}^{\mathsf{sid}}$ at random, since this is how HIC encryptions are formed. SIM services HIC encryption queries as $\mathcal{F}_{\mathsf{HIC}}$ does except that it collects the ciphertexts created by any encryption query and the ciphertexts chosen for every honest session in set Cset, and aborts if either process regenerates a

SIM interacts with environment $\mathcal{Z}$'s interface $\mathcal{A}$ and with functionality $\mathcal{F}_{\mathsf{pwKE}}$. W.l.o.g. we assume that $\mathcal{A}$ uses AdvDec to implement Dec queries to $\mathcal{F}_{\mathsf{HIC}}$.

Initialization: Set $\mathsf{Cset} = \{\}$, set $\mathsf{THIC}_{\hat{pw}}$ as an empty table and $\mathsf{c2pw}[c] := \perp$ for all values $\hat{pw}$ and c.

Notation (used in all security games in Figure 9)

Let $\mathsf{THIC}_{\hat{pw}}.\mathsf{s}[T]$ be a shortcut for set $\{s \in \mathcal{R} : \nexists \hat{m} \text{ s.t. } (\hat{m}, (s, T)) \in \mathsf{THIC}_{\hat{pw}}\}$.
Let $\mathsf{THIC}_{\hat{pw}}.\mathsf{c}$ be a shortcut for set $\{c \in \mathcal{D} : \nexists \hat{m} \text{ s.t. } (\hat{m}, c) \in \mathsf{THIC}_{\hat{pw}}\}$.
Let $\mathsf{THIC}_{\hat{pw}}.\mathsf{m}$ be a shortcut for set $\{m \in \mathcal{D} : \nexists \hat{c} \text{ s.t. } (m, \hat{c}) \in \mathsf{THIC}_{\hat{pw}}\}$.

On query $(\mathsf{NewSession}, \mathsf{sid}, \mathsf{P}, \mathsf{CP})$ from $\mathcal{F}_{\mathsf{pwKE}}$:

Set $\mathsf{fullsid} \leftarrow \mathsf{order}(\mathsf{sid}, \mathsf{P}, \mathsf{CP})$, $b \leftarrow \mathsf{bit}(\mathsf{P}, \mathsf{CP})$, $c \xleftarrow{r} \mathcal{D}$ (*abort* if $c \in \mathsf{Cset}$), add c to Cset, record $(\mathsf{sid}, \mathsf{P}, \mathsf{CP}, \mathsf{fullsid}, b, c)$, return c.

Emulating functionality $\mathcal{F}_{\mathsf{HIC}}$:

- On $\mathcal{A}$'s query $(\mathsf{Enc}, \hat{pw}, M)$ to $\mathcal{F}_{\mathsf{HIC}}$: Set $r \xleftarrow{r} \mathcal{R}$, $m \leftarrow (r, M)$. If $(m, c) \in \mathsf{THIC}_{\hat{pw}}$ return c; Else pick $c \xleftarrow{r} \mathsf{THIC}_{\hat{pw}}.\mathsf{c}$ (*abort* if $c \in \mathsf{Cset}$), set $\mathsf{c2pw}[c] \leftarrow \hat{pw}$, add c to Cset and (m, c) to $\mathsf{THIC}_{\hat{pw}}$, return c.
- On $\mathcal{A}$'s query $(\mathsf{AdvEnc}, \hat{pw}, m, T)$ to $\mathcal{F}_{\mathsf{HIC}}$: If $(m, c) \in \mathsf{THIC}_{\hat{pw}}$ return c; Else pick $s \xleftarrow{r} \mathsf{THIC}_{\hat{pw}}.\mathsf{s}[T]$, set $c \leftarrow (s, T)$ (*abort* if $c \in \mathsf{Cset}$), set $\mathsf{c2pw}[c] \leftarrow \hat{pw}$, add c to Cset and (m, c) to $\mathsf{THIC}_{\hat{pw}}$, return c.
- On $\mathcal{A}$'s query $(\mathsf{AdvDec}, \hat{pw}, c)$ to $\mathcal{F}_{\mathsf{HIC}}$: If $(m, c) \in \mathsf{THIC}_{\hat{pw}}$ return m; Else pick $r \xleftarrow{r} \mathcal{R}$ and $(x, M) \xleftarrow{r} \mathsf{KA.msg}$, set $m \leftarrow (r, M)$, add (m, c) to $\mathsf{THIC}_{\hat{pw}}$ (*abort* if $\exists\ \hat{c} \neq c$ s.t. $(m, \hat{c}) \in \mathsf{THIC}_{\hat{pw}}$), save $(\mathsf{backdoor}, c, \hat{pw}, x)$, return m.

On $\mathcal{A}$'s message $\hat{c}$ to session $\mathsf{P}^{\mathsf{sid}}$: (accept only the first such message)

Retrieve record $(\mathsf{sid}, \mathsf{P}, \mathsf{CP}, \mathsf{fullsid}, b, c)$ and do:

1. If there is record $(\mathsf{sid}, \mathsf{CP}, \mathsf{P}, \mathsf{fullsid}, \neg b, \hat{c})$: send $(\mathsf{NewKey}, \mathsf{sid}, \mathsf{P}, \perp)$ to $\mathcal{F}_{\mathsf{pwKE}}$;
2. Otherwise set $\hat{pw} \leftarrow \mathsf{c2pw}[\hat{c}]$ and do the following:
   (a) If $\hat{pw} = \perp$ or $\hat{pw} = (\hat{\mathsf{fullsid}}, \hat{b}, \cdot)$ for $(\hat{\mathsf{fullsid}}, \hat{b}) \neq (\mathsf{fullsid}, \neg b)$, send $(\mathsf{TestPwd}, \mathsf{sid}, \mathsf{P}, \perp)$ and $(\mathsf{NewKey}, \mathsf{sid}, \mathsf{P}, \perp)$ to $\mathcal{F}_{\mathsf{pwKE}}$;
   (b) If $\hat{pw} = (\mathsf{fullsid}, \neg b, pw^*)$ retrieve $((\hat{r}, \hat{M}), \hat{c})$ from $\mathsf{THIC}_{\hat{pw}}$ and:
      i. service $\mathcal{F}_{\mathsf{HIC}}$'s query $(\mathsf{AdvDec}, (\mathsf{fullsid}, b, pw^*), c)$, retrieve $(\mathsf{backdoor}, c, (\mathsf{fullsid}, b, pw^*), x)$;
      ii. set $K \leftarrow \mathsf{KA.key}(x, \hat{M})$, send $(\mathsf{TestPwd}, \mathsf{sid}, \mathsf{P}, pw^*)$ and $(\mathsf{NewKey}, \mathsf{sid}, \mathsf{P}, K)$ to $\mathcal{F}_{\mathsf{pwKE}}$.

Fig. 8: Simulator SIM for the proof of Theorem 2

ciphertext in Cset. Here we use the fact that even though an adversary can set the T part of the ciphertext $c = (s, T)$ resulting from an adversarial encryption query AdvEnc, the s part of c is chosen at random, and this prevents ciphertext collisions (except with negligible probability) if $|\mathcal{R}| \geq 2^{2\kappa}$. Hence, assuming that $\mathcal{R}$ is big enough, we have that (1) each adversarial ciphertext can be matched to (at most) one password on which it decrypts to a non-random value in space $\mathcal{M}$, and (2) the simulator can extract this unique password and retrieve the corresponding plaintext (SIM stores the key $\hat{pw}$ which was used to create ciphertext c in the c2pw table by setting $\mathsf{c2pw}[c] \leftarrow \hat{pw}$). Moreover, since by the same collision-resistant property of $\mathcal{F}_{\mathsf{HIC}}$ ciphertexts the adversary cannot "hit" any honest session $\mathsf{P}^{\mathsf{sid}}$'s ciphertext c via an encryption query, the decryption of $\mathsf{P}^{\mathsf{sid}}$'s ciphertext on each password is also a random value in $\mathcal{M}$. By the message-randomness property of KA, simulator SIM can embed messages of fresh KA instances into each decryption query, and combining this with fact (1) above allows for a reduction of EKE instances corresponding to "wrong" password guesses to the KA's security.

Let q_{IC} be the bound on the number of queries $\mathcal{Z}$ makes to the interfaces of the (randomized) ideal cipher $\mathcal{F}_{\mathsf{HIC}}$, and let q_P be the upper-bound on the number of honest EKE sessions $\mathsf{P}^{\mathsf{sid}}$ which $\mathcal{Z}$ invokes for any identifiers P, sid. [8] Let $\varepsilon_{\mathsf{KA.sec}}$ and $\varepsilon_{\mathsf{KA.rand}}$ be the upper-bounds on the distinguishing advantage against, respectively, the security and the random-message properties of the key exchange scheme KA (see Section 2) of an adversary whose computational resources are roughly those of an environment $\mathcal{Z}$ extended by execution of $q_{IC} + q_P$ instances of the key exchange scheme KA.[9]

For space-constraint reasons we defer the details of the game changes and reductions to the full version [30], but we show the code of all successive games in Figure 9. By the arguments for indistinguishability of successive games, the total distinguishing advantage of environment $\mathcal{Z}$ between the real-world and the ideal-world interaction is upper-bounded by the following expression, which sums up the bounds argued in the full proof, see [30]:

$$(q_{IC} + q_P)\left[\frac{1}{|\mathcal{R}|} \cdot \left\{2q_P + q_{IC} + 2 \cdot \frac{q_{IC} + q_P}{|\mathcal{M}|}\right\} + \varepsilon_{\mathsf{KA.rand}} + q_P \cdot \varepsilon_{\mathsf{KA.sec}}\right] \quad (2)$$

Since this quantity is negligible if $\mathcal{R} = \{0,1\}^n$ for $n = O(\kappa)$, it implies Theorem 2. □

Notes on Exact Security. The dominating factors are $(q_{IC} + q_P)^2/|\mathcal{R}|$ and $(q_{IC} + q_P) \cdot (\varepsilon_{\mathsf{KA.rand}} + q_P \cdot \varepsilon_{\mathsf{KA.sec}})$. The first factor is due to possible collisions in Half-Ideal Cipher, and it is unavoidable using an arbitrary HIC realization because it is the probability of generating the same ciphertext c as an encryption of two different KA instances under two different passwords, which would also form an explicit attack on the security of EKE (the adversary would

[8] We assume that $\mathcal{Z}$ invokes at most two sessions for any fixed identifier sid.

[9] This bound involves $q_{IC} + q_P$ instead of q_P key exchange instances because our reductions to KA security run KA.msg for each adversarial AdvDec query to $\mathcal{F}_{\mathsf{HIC}}$.

Game 0: real-world interaction

initialization

Initialize $\mathsf{Cset} = \{\}$ and $\forall\, \hat{pw}$ empty $\mathsf{THIC}_{\hat{pw}}$

on $(\mathsf{NewSession}, \mathsf{sid}, \mathsf{P}, \mathsf{CP}, pw)$ to P:
$\mathsf{fullsid} \leftarrow \mathsf{order}(\mathsf{sid}, \mathsf{P}, \mathsf{CP})$, $b \leftarrow \mathsf{bit}(\mathsf{P}, \mathsf{CP})$, $\hat{pw} \leftarrow (\mathsf{fullsid}, b, pw)$
$(x, M) \xleftarrow{r} \mathsf{KA.msg}$
$c \leftarrow \mathcal{F}_{\mathsf{HIC}}.\mathsf{Enc}(\hat{pw}, M)$
save $(\mathsf{sid}, \mathsf{P}, \mathsf{CP}, \mathsf{fullsid}, b, pw, x, c, \perp)$, output c

on message $\hat{c}$ to session $\mathsf{P}^{\mathsf{sid}}$ (accept only one):
if $\exists$ record $(\mathsf{sid}, \mathsf{P}, \mathsf{CP}, \mathsf{fullsid}, b, pw, x, \cdot, \perp)$:
  $(\hat{r}, \hat{M}) \leftarrow \mathcal{F}_{\mathsf{HIC}}.\mathsf{AdvDec}((\mathsf{fullsid}, \neg b, pw), \hat{c})$
  $K \leftarrow \mathsf{KA.key}(x, \hat{M})$ and output $(\mathsf{sid}, \mathsf{P}, K)$

on query $\mathcal{F}_{\mathsf{HIC}}.\mathsf{Enc}(\hat{pw}, M)$:
$r \xleftarrow{r} \mathcal{R}$, set $m \leftarrow (r, M)$
If $\exists\ c$ s.t. $(m, c) \in \mathsf{THIC}_{\hat{pw}}$:
  return c
else:
  pick $c \xleftarrow{r} \mathsf{THIC}_{\hat{pw}}.\mathsf{c}$,
  add c to Cset and (m, c) to $\mathsf{THIC}_{\hat{pw}}$
  return c

on query $\mathcal{F}_{\mathsf{HIC}}.\mathsf{AdvEnc}(\hat{pw}, m, T)$:
if $\exists\ c$ s.t. $(m, c) \in \mathsf{THIC}_{\hat{pw}}$:
  return c
else:
  $s \xleftarrow{r} \mathsf{THIC}_{\hat{pw}}.\mathsf{s}[T]$, set $c \leftarrow (s, T)$,
  add c to Cset and (m, c) to $\mathsf{THIC}_{\hat{pw}}$
  return c

on query $\mathcal{F}_{\mathsf{HIC}}.\mathsf{AdvDec}(\hat{pw}, c)$:
if $\exists\ m$ s.t. $(m, c) \in \mathsf{THIC}_{\hat{pw}}$:
  return m
else:
  $m \xleftarrow{r} \mathsf{THIC}_{\hat{pw}}.\mathsf{m}$, add (m, c) to $\mathsf{THIC}_{\hat{pw}}$
return m

Game 1: randomizing protocol communication

on $(\mathsf{NewSession}, \mathsf{sid}, \mathsf{P}, \mathsf{CP}, pw)$ to P:
set $(\mathsf{fullsid}, b, \hat{pw})$ as in Game 0
$(x, M) \xleftarrow{r} \mathsf{KA.msg}$, $r \xleftarrow{r} \mathcal{R}$, $c \xleftarrow{r} \mathcal{D}$
abort if $((r, M), *) \in \mathsf{THIC}_{\hat{pw}}$ or $c \in \mathsf{Cset}$
add $((r, M), c)$ to $\mathsf{THIC}_{\hat{pw}}$
save $(\mathsf{sid}, \mathsf{P}, \mathsf{CP}, \mathsf{fullsid}, b, pw, x, c, \perp)$, output c

Game 2: binding adversarial ciphertexts to passwords

on $\mathcal{F}_{\mathsf{HIC}}.\mathsf{Enc}(\hat{pw}, M)$ or $\mathcal{F}_{\mathsf{HIC}}.\mathsf{AdvEnc}(\hat{pw}, m, T)$:
Before adding c to Cset, do the following:
  abort if $c \in \mathsf{Cset}$
  set $\mathsf{c2pw}[c] \leftarrow \hat{pw}$

Game 3: adding trapdoors to decryption

on query $\mathcal{F}_{\mathsf{HIC}}.\mathsf{AdvDec}(\hat{pw}, c)$:

if $\exists m$ s.t. $(m, c) \in \mathsf{THIC}_{\hat{pw}}$ return m, otherwise:
  $(x, M) \xleftarrow{r} \mathsf{KA.msg}(1^\kappa)$, $r \xleftarrow{r} \mathcal{R}$, $m \leftarrow (r, M)$
  abort if $(m, *) \in \mathsf{THIC}_{\hat{pw}}$
  add (m, c) to $\mathsf{THIC}_{\hat{pw}}$
  save $(\mathsf{backdoor}, c, \hat{pw}, x)$, return m

Game 4: KA messages via AdvDec

on $(\mathsf{NewSession}, \mathsf{sid}, \mathsf{P}, \mathsf{CP}, pw)$ to P:
set $(\mathsf{fullsid}, b, \hat{pw})$ as in Game 0
$c \xleftarrow{r} \mathcal{D}$, *abort* if $c \in \mathsf{Cset}$, otherwise add c to Cset
query $\mathcal{F}_{\mathsf{HIC}}.\mathsf{AdvDec}(\hat{pw}, c)$
retrieve $(\mathsf{backdoor}, c, \hat{pw}, x)$
save $(\mathsf{sid}, \mathsf{P}, \mathsf{CP}, \mathsf{fullsid}, b, pw, x, c, \perp)$, output c

Game 5: extracting passwords

on message $\hat{c}$ to session $\mathsf{P}^{\mathsf{sid}}$:
if $\exists$ record $\mathsf{rec} = (\mathsf{sid}, \mathsf{P}, \mathsf{CP}, \mathsf{fullsid}, b, pw, x, c, \perp)$:
  if $\exists$ record $(\mathsf{sid}, \mathsf{CP}, \mathsf{P}, \mathsf{fullsid}, \neg b, pw, \cdot, \hat{c}, \hat{K})$
  s.t. $\mathcal{Z}$ sent c to $\mathsf{CP}^{\mathsf{sid}}$:
    $K \leftarrow \hat{K}$
  else:
    $\hat{pw} \leftarrow \mathsf{c2pw}[\hat{c}]$
    if $\hat{pw} = (\mathsf{fullsid}, \neg b, pw)$:
      retrieve $((\hat{r}, \hat{M}), \hat{c})$ from $\mathsf{THIC}_{\hat{pw}}$,
      set $K \leftarrow \mathsf{KA.key}(x, \hat{M})$
    else:
      $K \xleftarrow{r} \{0,1\}^\kappa$
  reset $\mathsf{rec} \leftarrow (\mathsf{sid}, \mathsf{P}, \mathsf{CP}, \mathsf{fullsid}, b, pw, x, c, K)$
  output $(\mathsf{sid}, \mathsf{P}, K)$

Game 6: delaying password usage

on $(\mathsf{NewSession}, \mathsf{sid}, \mathsf{P}, \mathsf{CP}, pw)$ to P:
$\mathsf{fullsid} \leftarrow \mathsf{order}(\mathsf{sid}, \mathsf{P}, \mathsf{CP})$, $b \leftarrow \mathsf{bit}(\mathsf{P}, \mathsf{CP})$
$c \xleftarrow{r} \mathcal{D}$, *abort* if $c \in \mathsf{Cset}$, otherwise add c to Cset
save $(\mathsf{sid}, \mathsf{P}, \mathsf{CP}, \mathsf{fullsid}, b, pw, \perp, c, \perp)$, output c

on message $\hat{c}$ to session $\mathsf{P}^{\mathsf{sid}}$:
if $\exists$ record $(\mathsf{sid}, \mathsf{P}, \mathsf{CP}, \mathsf{fullsid}, b, pw, \perp, c, \perp)$:
  if $\exists$ record $(\mathsf{sid}, \mathsf{CP}, \mathsf{P}, \mathsf{fullsid}, \neg b, pw, \perp, \hat{c}, \hat{K})$:
    $K \leftarrow \hat{K}$
  else:
    $\hat{pw} \leftarrow \mathsf{c2pw}[\hat{c}]$
    if $\hat{pw} = (\mathsf{fullsid}, \neg b, pw)$:
      query $\mathcal{F}_{\mathsf{HIC}}.\mathsf{AdvDec}((\mathsf{fullsid}, b, pw), c)$,
      retrieve $(\mathsf{backdoor}, c, \cdot, x)$
      retrieve $((\hat{r}, \hat{M}), \hat{c})$ from $\mathsf{THIC}_{\hat{pw}}$,
      set $K \leftarrow \mathsf{KA.key}(x, \hat{M})$
    else:
      $K \xleftarrow{r} \{0,1\}^\kappa$
  reset $\mathsf{rec} \leftarrow (\mathsf{sid}, \mathsf{P}, \mathsf{CP}, \mathsf{fullsid}, b, pw, x, c, K)$
  output $(\mathsf{sid}, \mathsf{P}, K)$

Fig. 9: Game changes for the proof of Theorem 2 (compare Fig. 8 for notation)

effectively make two password guesses in one on-line interaction). However, whereas the bound $(q_{IC})^2/|\mathcal{R}|$ is tight if the encryption is modeled as a Half-Ideal Cipher, we do not know if it is tight in relation to the specific modified 2-Feistel instantiation of Half-Ideal Cipher, because we do not know how to stage an explicit attack on EKE using modified 2-Feistel along these lines. This relates to the fact that whereas the modified 2-Feistel realizes functionality $\mathcal{F}_{\mathsf{HIC}}$, this functionality allows more freedom to the adversary than the modified 2-Feistel construction. Namely, whereas $\mathcal{F}_{\mathsf{HIC}}$ allows the adversary to encrypt any messages M using a ciphertext $c = (s, T)$ where T can be freely set, the same is not true about the modified 2-Feistel construction, where for any fixed M the adversary can choose T from the set of values of the form $T = M/\mathsf{H}(pw, r)$ for some r.

The second factor is due to reductions to KA security properties. Note that some KA schemes, e.g. Diffie-Hellman, have perfect message-randomness, i.e. $\varepsilon_{\mathsf{KA.rand}} = 0$. Further, if the KA scheme is *random self-reducible*, as is Diffie-Hellman, then this factor can be reduced to $\varepsilon_{\mathsf{KA.sec}}$ because a reduction to KA security for the transition between Games 4 and 5, see the proof in [30], can then be modified so that it deals with all honest sessions at once instead of staging a hybrid argument over all sessions, and it embeds randomized versions of the KA challenge into each decryption query rather than guessing a target query.

5.1 EKE with Half-Ideal Cipher: the KEM version

In Figure 10 we show protocol EKE-KEM, which is a KEM version of the EKE protocol using a Half-Ideal Cipher. In the 1-flow protocol EKE considered in Figure 7, the message flows are generated by a single-round KA scheme, whereas here we consider an EKE variant which is built from any two-flow key exchange, i.e. KEM, see Section 2.2. The drawback is that it is 2-flow instead of 1-flow, but the benefits are that the HIC can be used only for one message, so if KEM is instantiated with Diffie-Hellman and HIC is implemented using m2F, this implies a single RO hash onto a group per party instead of two such hashes. Moreover, this version of EKE can use any CPA-secure KEM as a black box, as long as the KEM satisfies the anonymity and uniform public keys properties, which implies, e.g., lattice-based UC PAKE given any lattice-based KEM with these properties.

Note that in the protocol of Fig. 10 party P_0 outputs a random session key if the key confirmation message τ fails to verify. This is done only so that the protocol conforms to the implicit-authentication functionality $\mathcal{F}_{\mathsf{pwKE}}$. In practice P_0 could output $\perp$ in this case, and this would implement explicit authentication in the P_1-to-P_0 direction.

Theorem 3. *If* KEM *is IND secure, anonymous, and has uniform public keys in domain $\mathcal{PK}$ (see Section 2.2),* HIC *is a UC Half-Ideal Cipher in domain $\mathcal{R} \times \mathcal{PK}$, and* H *is an RO hash, then protocol* EKE-KEM *realizes the UC PAKE functionality $\mathcal{F}_{\mathsf{pwKE}}$.*

The proof of Theorem 3 is deferred to the full version [30]. It follows the same blueprint as the proof of Theorem 2. The most important intuition needed

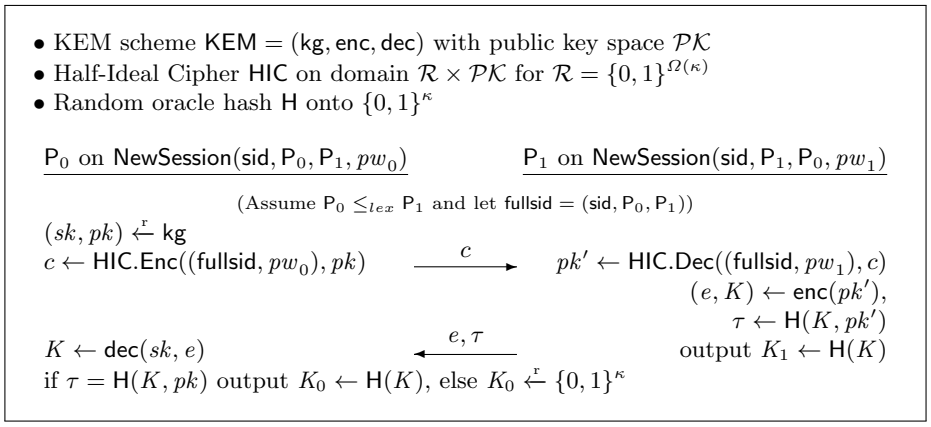

Fig. 10: EKE-KEM: Encrypted Key Exchange with Half-Ideal Cipher (KEM version)

for the adaptation of the proof of Theorem 2 to the proof of Theorem 3 is why it works for KEMs that satisfy the anonymity property: The key issue is that we need anonymity of the KEM ciphertext e only for honest keys pk and not for adversarial ones, and the reason for this is that the only non-random pk under which an honest party encrypts is the key pk decrypted under a unique password guess pw^* used in the adversarial ciphertext c this party receives. If pw^* equals to P_1's password pw then this session is already successfully attacked, so the non-randomness of P_1's ciphertext is not an issue. But if $pw^* \neq pw$ then KEM ciphertext e is effectively encrypted under key $pk' = \mathsf{AdvDec}(pw, c)$ which is random, and the key confirmation works as a commitment to the KEM key pk decrypted from HIC ciphertext c, hence also to the password used in that decryption. This commitment is also effectively encrypted under the KEM session key K, hence it can be verified only by a party which created pk and HIC-encrypted it under the right pw. Here we again rely on the property of HIC, which just like IC assures that decryption under any password except for the unique password committed in the ciphertext results in a random plaintext, i.e. a random KEM public key pk, which makes the KEM session key K encrypted under such pk hidden to the adversary by KEM security.

We note that the key confirmation could involve directly pw instead of pk, but pk is a commitment to pw unless the adversary creates a collision in HIC plaintext, and using pk instead of pw lets P_0 erase pw after sending its first message. This way an adaptive compromise on party P_0 during protocol execution allows for offline dictionary attack on the password, but does not leak it straight away. (Note that adaptive party compromise is not part of our security model.) We note also that RO hash H can probably be replaced by a key derivation function which is both a CRH (because it needs to commit to pk) and a PRF (because it must encrypt this commitment under K), but since HIC implies RO hash (and indeed our m2Fuses it) we opt for the simpler option of RO hash to compute the authenticator.

6 Applications of Half-Ideal Cipher to aPAKE

Gu et al. [33] proposed an asymmetric PAKE protocol called KHAPE which is a generic compiler from any UC *key-hiding* Authenticated Key Exchange (AKE), using an Ideal Cipher on the domain formed by (private, public) key pairs of the AKE. We show that KHAPE realizes UC aPAKE if IC is replaced by HIC. For lack of space the proof of the following Theorem is deferred to the full version [30]. For reference, for AKE functionality $\mathcal{F}_{\mathsf{khAKE}}$ see e.g., [33], and for aPAKE functionality $\mathcal{F}_{\mathsf{aPAKE}}$ see e.g., [30].

Theorem 4. *Protocol* KHAPE *of [33] realizes the UC aPAKE functionality* $\mathcal{F}_{\mathsf{aPAKE}}$ *if the AKE protocol realizes the Key-Hiding AKE functionality* $\mathcal{F}_{\mathsf{khAKE}}$ *assuming that* kdf *is a secure PRF and* HIC *is a half-ideal cipher over message space of private and public key pairs in AKE.*

We note that Freitas et al. [31] showed a UC aPAKE which improves upon protocol KHAPE of [33] in round complexity. The aPAKE of [31] relies on IC in a similar way as protocol KHAPE, and the proof therein should also generalize to the case when IC is replaced by HIC.

References

1. Abdalla, M., Catalano, D., Chevalier, C., Pointcheval, D.: Efficient two-party password-based key exchange protocols in the UC framework. In: Malkin, T. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 335–351. Springer, Heidelberg (Apr 2008). https://doi.org/10.1007/978-3-540-79263-5_22
2. Albrecht, M.R., Bernstein, D.J., Chou, T., Cid, C., Gilcher, J., Lange, T., Maram, V., von Maurich, I., Misoczki, R., Niederhagen, R., Paterson, K.G., Persichetti, E., Peters, C., Schwabe, P., Sendrier, N., Szefer, J., Tjhai, C.J., Tomlinson, M., Wang, W.: Classic mceliece: Nist round 3 submission, https://csrc.nist.gov/Projects/post-quantum-cryptography/round-3-submissions (2021)
3. Alkim, .E., Bos, J.W., Ducas, L., Longa, P., Mironov, I., Naehrig, M., Nikolaenko, V., Peikert, C., Raghunathan, A., Stebil, D.: Frodokem: Nist round 3 submission, https://csrc.nist.gov/Projects/post-quantum-cryptography/round-3-submissions (2021)
4. Andreeva, E., Bogdanov, A., Dodis, Y., Mennink, B., Steinberger, J.P.: On the indifferentiability of key-alternating ciphers. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 531–550. Springer, Heidelberg (Aug 2013). https://doi.org/10.1007/978-3-642-40041-4_29
5. Bellare, M., Boldyreva, A., Desai, A., Pointcheval, D.: Key-privacy in public-key encryption. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 566–582. Springer, Heidelberg (Dec 2001). https://doi.org/10.1007/3-540-45682-1_33
6. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Advances in Cryptology – EUROCRYPT 2000. pp. 139–155. Springer (2000)
7. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Denning, D.E., Pyle, R., Ganesan, R., Sandhu, R.S., Ashby, V. (eds.) ACM CCS 93. pp. 62–73. ACM Press (Nov 1993). https://doi.org/10.1145/168588.168596

8. Bellovin, S.M., Merritt, M.: Encrypted key exchange: Password-based protocols secure against dictionary attacks. In: IEEE Computer Society Symposium on Research in Security and Privacy – S&P 1992. pp. 72–84. IEEE (1992)
9. Bernstein, D.J., Hamburg, M., Krasnova, A., Lange, T.: Elligator: elliptic-curve points indistinguishable from uniform random strings. In: Sadeghi, A.R., Gligor, V.D., Yung, M. (eds.) ACM CCS 2013. pp. 967–980. ACM Press (Nov 2013). https://doi.org/10.1145/2508859.2516734
10. Bernstein, D.J., Kölbl, S., Lucks, S., Massolino, P.M.C., Mendel, F., Nawaz, K., Schneider, T., Schwabe, P., Standaert, F.X., Todo, Y., Viguier, B.: Gimli : A cross-platform permutation. In: Fischer, W., Homma, N. (eds.) CHES 2017. LNCS, vol. 10529, pp. 299–320. Springer, Heidelberg (Sep 2017). https://doi.org/10.1007/978-3-319-66787-4_15
11. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: Keccak. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 313–314. Springer, Heidelberg (May 2013). https://doi.org/10.1007/978-3-642-38348-9_19
12. Black, J., Rogaway, P.: Ciphers with arbitrary finite domains. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 114–130. Springer, Heidelberg (Feb 2002). https://doi.org/10.1007/3-540-45760-7_9
13. Black, J., Rogaway, P., Shrimpton, T.: Black-box analysis of the block-cipher-based hash-function constructions from PGV. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 320–335. Springer, Heidelberg (Aug 2002). https://doi.org/10.1007/3-540-45708-9_21
14. Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehle, D.: Crystals - kyber: A cca-secure module-lattice-based kem. In: 2018 IEEE European Symposium on Security and Privacy (EuroS P). pp. 353–367 (2018). https://doi.org/10.1109/EuroSP.2018.00032
15. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: IEEE Symposium on Foundations of Computer Science – FOCS 2001. pp. 136–145. IEEE (2001)
16. Canetti, R., Halevi, S., Katz, J., Lindell, Y., MacKenzie, P.: Universally composable password-based key exchange. In: Advances in Cryptology – EUROCRYPT 2005. pp. 404–421. Springer (2005)
17. Coron, J.S., Dodis, Y., Mandal, A., Seurin, Y.: A domain extender for the ideal cipher. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 273–289. Springer, Heidelberg (Feb 2010). https://doi.org/10.1007/978-3-642-11799-2_17
18. Coron, J.S., Patarin, J., Seurin, Y.: The random oracle model and the ideal cipher model are equivalent. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 1–20. Springer, Heidelberg (Aug 2008). https://doi.org/10.1007/978-3-540-85174-5_1
19. Dachman-Soled, D., Katz, J., Thiruvengadam, A.: 10-round Feistel is indifferentiable from an ideal cipher. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 649–678. Springer, Heidelberg (May 2016). https://doi.org/10.1007/978-3-662-49896-5_23
20. Daemen, J., Hoffert, S., Assche, G.V., Keer, R.V.: The design of Xoodoo and Xoofff. IACR Trans. Symm. Cryptol. **2018**(4), 1–38 (2018). https://doi.org/10.13154/tosc.v2018.i4.1-38
21. Dai, Y., Seurin, Y., Steinberger, J.P., Thiruvengadam, A.: Indifferentiability of iterated Even-Mansour ciphers with non-idealized key-schedules: Five rounds are necessary and sufficient. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part III. LNCS, vol. 10403, pp. 524–555. Springer, Heidelberg (Aug 2017). https://doi.org/10.1007/978-3-319-63697-9_18

22. Dai, Y., Steinberger, J.P.: Indifferentiability of 8-round Feistel networks. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part I. LNCS, vol. 9814, pp. 95–120. Springer, Heidelberg (Aug 2016). https://doi.org/10.1007/978-3-662-53018-4_4
23. D'Anvers, J.P., Karmakar, A., Sinha Roy, S., Vercauteren, F.: Saber: Module-lwr based key exchange, cpa-secure encryption and cca-secure kem. In: Joux, A., Nitaj, A., Rachidi, T. (eds.) Progress in Cryptology – AFRICACRYPT 2018. pp. 282–305. Springer International Publishing, Cham (2018)
24. Desai, A.: The security of all-or-nothing encryption: Protecting against exhaustive key search. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 359–375. Springer, Heidelberg (Aug 2000). https://doi.org/10.1007/3-540-44598-6_23
25. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Transactions on Information Theory **22**(6), 644–654 (1976)
26. Dodis, Y., Puniya, P.: Feistel networks made public, and applications. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 534–554. Springer, Heidelberg (May 2007). https://doi.org/10.1007/978-3-540-72540-4_31
27. Dodis, Y., Stam, M., Steinberger, J.P., Liu, T.: Indifferentiability of confusion-diffusion networks. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 679–704. Springer, Heidelberg (May 2016). https://doi.org/10.1007/978-3-662-49896-5_24
28. Even, S., Mansour, Y.: A construction of a cipher from a single pseudorandom permutation. In: Imai, H., Rivest, R.L., Matsumoto, T. (eds.) ASIACRYPT'91. LNCS, vol. 739, pp. 210–224. Springer, Heidelberg (Nov 1993). https://doi.org/10.1007/3-540-57332-1_17
29. Faz-Hernandez, A., Scott, S., Sullivan, N., Wahby, R., Wood, C.: Hashing to elliptic curves, irft-cfrg active draft, https://datatracker.ietf.org/doc/draft-irtf-cfrg-hash-to-curve/ (2022)
30. Freitas Dos Santos, B., Gu, Y., Jarecki, S.: Randomized half-ideal cipher on groups with applications to UC (a)PAKE. Cryptology ePrint Archive, Report 2023/295 (2023), http://eprint.iacr.org/2023/295
31. Freitas Dos Santos, B., Gu, Y., Jarecki, S., Krawczyk, H.: Asymmetric PAKE with low computation and communication. In: EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer (2022)
32. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M.J. (ed.) CRYPTO'99. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (Aug 1999). https://doi.org/10.1007/3-540-48405-1_34
33. Gu, Y., Jarecki, S., Krawczyk, H.: KHAPE: Asymmetric PAKE from key-hiding key exchange. In: Advances in Cryptology - Crypto 2021. pp. 701–730 (2021), https://ia.cr/2021/873
34. Guo, C., Lin, D.: Improved domain extender for the ideal cipher. Cryptography Commun. **7**(4), 509–533 (dec 2015). https://doi.org/10.1007/s12095-015-0128-7, https://doi.org/10.1007/s12095-015-0128-7
35. Hoffstein, J., Pipher, J., Silverman, J.H.: Ntru: A ring-based public key cryptosystem. In: Buhler, J.P. (ed.) Algorithmic Number Theory. pp. 267–288. Springer Berlin Heidelberg, Berlin, Heidelberg (1998)
36. Holenstein, T., Künzler, R., Tessaro, S.: The equivalence of the random oracle model and the ideal cipher model, revisited. In: Fortnow, L., Vadhan, S.P. (eds.) 43rd ACM STOC. pp. 89–98. ACM Press (Jun 2011). https://doi.org/10.1145/1993636.1993650

37. Jaulmes, É., Joux, A., Valette, F.: On the security of randomized CBC-MAC beyond the birthday paradox limit: A new construction. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 237–251. Springer, Heidelberg (Feb 2002). https://doi.org/10.1007/3-540-45661-9_19
38. Kilian, J., Rogaway, P.: How to protect DES against exhaustive key search. In: Koblitz, N. (ed.) CRYPTO'96. LNCS, vol. 1109, pp. 252–267. Springer, Heidelberg (Aug 1996). https://doi.org/10.1007/3-540-68697-5_20
39. Kim, T., Tibouchi, M.: Invalid curve attacks in a GLS setting. In: Tanaka, K., Suga, Y. (eds.) IWSEC 15. LNCS, vol. 9241, pp. 41–55. Springer, Heidelberg (Aug 2015). https://doi.org/10.1007/978-3-319-22425-1_3
40. Krawczyk, H.: HMQV: A high-performance secure Diffie-Hellman protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (Aug 2005). https://doi.org/10.1007/11535218_33
41. Maram, V., Grubbs, P., Paterson, K.G.: Anonymous, robust post-quantum public key encryption. In: EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer (2022)
42. Marlinspike, M., Perrin, T.: The X3DH key agreement protocol, https://signal.org/docs/specifications/x3dh/ (2016)
43. McQuoid, I., Rosulek, M., Roy, L.: Minimal symmetric PAKE and 1-out-of-n OT from programmable-once public functions. In: CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020. (2020). https://doi.org/10.1145/3372297.3417870, https://eprint.iacr.org/2020/1043
44. Merkle, R.C.: One way hash functions and DES. In: Brassard, G. (ed.) CRYPTO'89. LNCS, vol. 435, pp. 428–446. Springer, Heidelberg (Aug 1990). https://doi.org/10.1007/0-387-34805-0_40
45. Preneel, B., Govaerts, R., Vandewalle, J.: Hash functions based on block ciphers: A synthetic approach. In: Stinson, D.R. (ed.) CRYPTO'93. LNCS, vol. 773, pp. 368–378. Springer, Heidelberg (Aug 1994). https://doi.org/10.1007/3-540-48329-2_31
46. Shannon, C.E.: Communication theory of secrecy systems. The Bell System Technical Journal **28**(4), 656–715 (1949). https://doi.org/10.1002/j.1538-7305.1949.tb00928.x
47. Tibouchi, M.: Elligator squared: Uniform points on elliptic curves of prime order as uniform random strings. In: Christin, N., Safavi-Naini, R. (eds.) FC 2014. LNCS, vol. 8437, pp. 139–156. Springer, Heidelberg (Mar 2014). https://doi.org/10.1007/978-3-662-45472-5_10
48. Winternitz, R.S.: Producing a one-way hash function from DES. In: Chaum, D. (ed.) CRYPTO'83. pp. 203–207. Plenum Press, New York, USA (1983)
49. Xagawa, K.: Anonymity of NIST PQC round 3 KEMs. In: EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer (2022)