Improving the Security of MACs via Randomized Message Preprocessing

Yevgeniy Dodis[1] and Krzysztof Pietrzak[2]

[1] New York University, Email: `dodis@cs.nyu.edu`
[2] CWI Amsterdam, Email: `k.z.pietrzak@cwi.nl`

Abstract. "Hash then encrypt" is an approach to message authentication, where first the message is hashed down using an ε-universal hash function, and then the resulting k-bit value is encrypted, say with a block-cipher. The security of this scheme is proportional to εq^2, where q is the number of MACs the adversary can request. As ε is at least 2^{-k}, the best one can hope for is $O(q^2/2^k)$ security. Unfortunately, such small ε is not achieved by simple hash functions used in practice, such as the polynomial evaluation or the Merkle-Damgård construction, where ε grows with the message length L.
The main insight of this work comes from the fact that, by using *randomized message preprocessing* via a short random salt p (which must then be sent as part of the authentication tag), we can use the "hash then encrypt" paradigm with suboptimal "practical" ε-universal hash functions, and still improve its exact security to optimal $O(q^2/2^k)$. Specifically, by using at most an $O(\log L)$-bit salt p, one can *always* regain the optimal exact security $O(q^2/2^k)$, even in situations where ε grows polynomially with L. We also give very simple preprocessing maps for popular "suboptimal" hash functions, namely polynomial evaluation and the Merkle-Damgård construction.
Our results come from a general extension of the classical Carter-Wegman paradigm, which we believe is of independent interest. On a high level, it shows that public randomization allows one to use the potentially much smaller "average-case" collision probability in place of the "worst-case" collision probability ε.

1 Introduction

Hash then Encrypt. A popular paradigm to message authentication is "hash then encrypt", where the authentication tag for a message m is computed as $f(h(m))$ where h is a hash function and f a pseudorandom permutation (say AES). This approach is appealing for several reasons: (1) it is stateless, (2) h needs not to be a cryptographic hash function, but only ε-universal[3] and (3) the "slow" cryptographic f is then only applied on a short input (i.e. on the range of h).

[3] A family of hash functions $\mathcal{H}$ is ε-universal, if for any $x \neq x'$, $\mathsf{Pr}_{h \leftarrow \mathcal{H}}[h(x) = h(x')] \leq \varepsilon$.

As f is indistinguishable from a uniformly random permutation, everything an attacker learns about $h(m_1), h(m_2), \ldots$ from the authentication tags $f(h(m_1))$, $f(h(m_2)), \ldots$ is whether there is a collision (as $f(h(m_i)) = f(h(m_j))$ iff $h(m_i) = h(m_j)$). Since h is not cryptographic, finding such a collision is usually enough for an adversary to come up with a forgery for a new message. If f is over $\{0,1\}^k$ (say, f is AES-128 where $k = 128$) then by the birthday bound the best security we can hope for is something in the order of $q^2/2^k$ where $q = q_{mac} + q_{forge}$ is the number of MAC queries and forgery attempts the adversary is allowed to make. More precisely, assuming that f is ideal (i.e. a uniform random permutation), the probability of a successful forgery can be upper bounded by[4]

$$\varepsilon \cdot q_{mac}^2 + \varepsilon \cdot q_{forge}$$

As $\varepsilon \geq 1/2^k$ the best one can hope for by the "hash then encrypt" approach is $O(q^2/2^k)$ security. In the sequel, we will call this *optimal* security.

Using Suboptimal Hash Functions. Unfortunately, hash functions used in practice, such as polynomial evaluation and the cascade (aka Merkle-Damgård) construction[5], do not yield optimal security, since the value ε they achieve grows linearly with the length of the message. There are several ways to improve the exact security with such hash functions. Most obviously, one can increase the security parameter k. However, this does not regain the optimal security relative to this *larger* k (although the absolute exact security is improved). More critically, increasing k is typically not an option in practice, since k is tied to the block length of the block cipher which is usually fixed (say, to 128 bits for AES) and pretty inflexible to any changes.[6] Another option is to design a different ε-universal hash function achieving optimal $\varepsilon = O(1/2^k)$. For one thing, replacing existing and popular implementations is not so easy in practice, so this option is usually ruled out anyway. More importantly, and this is part of the reason practical hash functions are not "optimally universal", $O(1/2^k)$-universal hash functions tend to be much less efficient or convenient than their slightly suboptimal counter-parts. For example, achieving perfect $\varepsilon = 1/2^k$ requires a key of size L [23], where L is the length of the message, which is very impractical. In theory, one can achieve $\varepsilon = 2/2^k$ by composing a "practical" δ-universal hash function from L to $k + \ell$ bits, where ℓ is chosen large enough to bring δ below $1/2^k$ (typically $\delta = O(L)/2^{k+\ell}$ which gives $\ell = \log(L) + O(1)$), with the perfectly universal hash function from $k + \ell$ to k bits, whose key is then only $k + \ell$

[4] If f is not ideal but a pseudorandom permutation, then there should also be a term counting the insecurity of f. However, since this term is always the same and is independent of the hash component, we will omit it from all our bounds.

[5] This construction uses a fixed input length shrinking function iteratively in order to get a function for arbitrary long inputs. In this paper we always assume that the iterated function is ideal, i.e. a uniformly random function.

[6] Though, if one is willing to make two invocations to the block-cipher, one can use it in CBC-mode in order to get a $\{0,1\}^{2k} \rightarrow \{0,1\}^k$ PRF, which can then be used instead of the block-cipher. This mode of operation should not be called "hash then encrypt", as the application of the (shrinking) PRF is not invertible, i.e. not an encryption scheme.

bits long. In practice, however, this composition is quite inconvenient to implement. Aside from the obvious inefficiency that the key size is at least doubled compared to using practical hash functions, the latter are usually optimized to operate on a specific value of k, and do not extend easily to larger outputs $k+\ell$, even if ℓ is very small. For example, the polynomial evaluation function with $k = 128$ corresponds to performing fast field operations over $GF(2^{128})$, which could be implemented in hardware. In contrast, evaluating field operations over $GF(2^{128+\ell})$ would be much slower even for $\ell = 1$, since a 129-bit element would not fit into a register. Similarly, the cascade construction typically uses a very specific compression function with fixed k, which is usually undefined for larger k.

To summarize, in practice the "hash then encrypt" paradigm does not achieve optimal exact security $O(q^2/2^k)$.

The question addressed in this paper is whether one can reclaim — with little extra cost — the optimal exact security of the "hash then encrypt" MAC when using such popular but sub-optimal ε-universal hash functions. Moreover, we would like a solution which does not change the value of k and does not modify the internals of the underlying hash function, so that our solution can be easily applied to existing implementations. Finally, the solution should remain stateless. Quite surprisingly, we answer this question in the affirmative for the popular polynomial evaluation and the cascade constructions, by using randomized message preprocessing before applying the actual MAC. We motivate our approach below.

Using Randomization. Recall, a hash function is ε-universal if the collision probability of two messages is at most ε *for all* possible message pairs. But the actual collision probability could be much smaller for most pairs. To illustrate this on an example, let us consider the polynomial-based ε-universal hash function: here the message $m \in \{0,1\}^{Lk}$ is viewed as a degree $(L-1)$ polynomial f_m over $GF(2^k)$, the secret key s is an element of $GF(2^k)$, and $h_s(m)$ is the value of the polynomial f_m at s. Given two distinct messages m_1 and m_2, their probability of collision is $r/2^k$, where r is the number of roots of the non-zero polynomial $g = f_{m_1} - f_{m_2}$. Although g can have up to $(L-1)$ roots for some pairs (m_1, m_2), Hartman and Raz [11] showed that the fraction of polynomials (of any degree) with t roots decays proportional to $1/t!$, which means that a vast majority of polynomials have at most a constant number of roots. Thus, a vast majority of pairs (m_1, m_2) have collision probability $O(1/2^k)$ rather than the worst-case bound $(L-1)/2^k$.

This brings up the following idea to improve the security of a MAC based on the "hash then encrypt" paradigm: apply some randomized preprocessing function $m' = Pre(m, p)$ to the message m (where p is a fresh random salt), and return $(p, f(h_s(m')))$ as the randomized MAC of m. The hope is that preprocessing should thwart the attempts of the adversary to choose messages which have a large collision probability and thus increase the security of the MAC. To put it differently, by designing a clever randomized message preprocessing, we will try to ensure that two *processed* messages cannot collide with probability more than $O(1/2^k)$, even if the worst-case ε is much larger. Another advantage of preprocessing comes from the fact that the original "suboptimal" MAC is used in a "black-box" manner, while suddenly becoming more secure!

Of course, there is a price we need to pay for regaining optimal exact security: we need to tell the receiving party how we randomized the message. Thus, aside from showing that randomized message preprocessing *can always* yield optimal exact security, — which is a non-obvious statement we will prove later, — the secondary objective is to minimize the number of random bits needed to avoid "bad" message pairs in a given ε-universal family.

The Salted Hash Function Paradigm. To capture the intuition just described, we introduce the concept of *salted* $(\varepsilon_{forge}, \varepsilon_{mac})$-universal hash functions. As the name suggests, the key of such a function has two parts, a "secret" part and a "public" salt. If the salt is empty, then $\varepsilon_{forge} = \varepsilon_{mac}$ and we have a usual ε-universal hash function with $\varepsilon = \varepsilon_{forge}$. Here ε_{forge} is an upper bound on the probability (over a choice of the secret key) that the hash values of two messages collide when the adversary can choose the messages and the public salt for both hashes, and ε_{mac} is the same probability but where at least one of the two public salts is chosen at random (clearly we always have $\varepsilon_{forge} \geq \varepsilon_{mac}$).

From such a salted hash function we can construct a MAC-scheme like from the usual ε-universal hash function (i.e. "hash then encrypt"), but where for every message to be authenticated the public salt is chosen at random and must be sent as a part of the authentication tag. As our first result, we generalize the standard "one-key" hash then encrypt MAC and show that the generalized MAC has security

$$\varepsilon_{mac} \cdot q_{mac}^2 + \varepsilon_{forge} \cdot q_{forge}$$

In particular, to get optimal $O(q^2/2^k)$ security here it is sufficient to have ε_{mac} in the order of $1/2^k$, while ε_{forge} can be considerably larger.

We also remark that the above salted hash paradigm is strictly more general than the "randomized preprocessing paradigm" advertised earlier. In particular, the latter corresponds to the salted hash functions of the form $h_s(Pre(m, p))$, where $Pre(m, p)$ is the preprocessing map. However, we find it more intuitive to state some of results for the general salted hash paradigm (which also makes them more general).

Salted Hash Functions with Short Salt. In this paper we propose randomized preprocessing mechanisms for several ε-universal hash functions and show that this turns them into $(\varepsilon_{forge}, \varepsilon_{mac})$-universal ones with $\varepsilon_{forge} \approx \varepsilon$ and $\varepsilon_{mac} = O(1/2^k)$. Moreover, in each case we use a very short public salt p.

Our first result is very general. In §4 we show (non-constructively) that for every balanced ε-universal hash function (where $\varepsilon = \varepsilon(L)$ can grow polynomially in the length of the messages L; i.e., $\varepsilon(L) = L^c/2^k$ for some constant c) there is a randomized preprocessing *using only* $O(\log L)$ *random bits* which gives an $(\varepsilon_{forge}, \varepsilon_{mac})$-universal hash function with $\varepsilon_{forge} \approx \varepsilon$ and $\varepsilon_{mac} = O(1/2^k)$. Although this result is non-constructive (i.e., the preprocessing map is inefficient and is only shown to exist), we believe the result is interesting given its generality.

Our next results involve simple and *efficient* implementations of the above generic result for popular ε-universal hash functions, such as the polynomial evaluation (denoted poly in the sequel) and the cascade (aka. Merkle-Damgård) construction. We describe these results in more details in §5 and §6, but mention that in each case we manage to design extremely simple preprocessing maps using an $O(\log L)$-bit public salt promised by the generic existence result: roughly, in the cascade construction we simply append a random string, and for poly the map consists of prepending a string chosen from a small set. Unfortunately the construction for poly is only non-uniform, i.e. we prove that maps with a succinct description (of length $O(L^3)$) exist, but do not give a specific map that works. Fortunately, we can show that almost all such succinct descriptions yield a good map, so one can just sample and "hardwire" a random string which will then define a good map almost certainly.

As our final result in §7, for hash functions which satisfy the slightly stronger property of being ε-Δ universal [16], we also show a *constructive* general result stating that we can always regain the optimal security by doing $O(\log L)$-bit *post*processing instead of preprocessing. By this we mean randomizing the hash value $h_s(m)$, rather than the message m, with an $O(\log L)$-bit public salt p. Moreover, we do not need the hash function to be balanced (which is necessary for the general result with preprocessing).

Figure 1 summarizes our results (see future sections for some of the notation). In Section §8 we give a numerical example to illustrate what security can be gained by using "salted" hash then encrypt. In Section §9 we review some related work and give some open problems.

2 Notation and Basic Definitions

For any integer $x \geq 0$ we denote by $\langle x \rangle_k$ its binary representation padded with leading 0's to length k, e.g. $\langle 7 \rangle_5 = 00111$. For two strings A, B we denote with $A \| B$ their concatenation. For $k \in \mathbb{N}$ we define $B_k \stackrel{\text{def}}{=} \{0,1\}^k$ and $B_k^{\leq i} \stackrel{\text{def}}{=} \cup_{j=1\ldots i} B_k^j$ to be the set of all strings of length at most i k-bit blocks. For a set $\mathcal{X}$ we denote with $x \leftarrow \mathcal{X}$ that x is sampled uniformly at random from $\mathcal{X}$.

Definition 1 (ε-almost 2-universal hash function) *A hash function $H : \mathcal{S} \times \mathcal{X} \to \mathcal{Y}$ is ε-almost 2-universal if for all $x, y \in \mathcal{X}$,$x \neq y$ and $h_s(.) \stackrel{\text{def}}{=} H(s, .)$*

$$\mathsf{Pr}[s \leftarrow \mathcal{S}; h_s(x) = h_s(y)] \leq \varepsilon$$

To save on notation we only write "ε-universal" for "ε-almost 2-universal". We also often consider the case where $\varepsilon = \varepsilon(\mathcal{X})$ can be a function of the message space $\mathcal{X}$.

3 Salted Hashing and MACs

In this section we first review the "hash then encrypt" approach for message authentication. We then define *salted universal hash functions*, which are a randomized version of normal ε-universal hash functions. Based on such hash functions, we propose a randomized version of "hash then encrypt" and show that

Construction	ε		length hash key	length of public salt	domain
	ε_{forge}	ε_{mac}			
$(\varepsilon_{forge}, \varepsilon_{mac})$-universal hash functions from ε-universal ones					
Generic: for each balanced $H(.)$ as below there is a permutation $g(.)$ such that					
$H(.)$	$L^c/2^k$		key for H	–	$\{0,1\}^L$
⋆ $H(g(.))$	$(L+\ell)^c/2^k$	$2/2^k$	key for H	$\ell \approx (2c+1)\log(L)$	$\{0,1\}^L$
Construction based on polynomial evaluation over $GF(2^k)$					
poly	$(L-1)/2^k$		k	–	$(\{0,1\}^k)^{\leq L}$
$\mathsf{poly}^{GF(2^k)}$	$L/2^k$	$1/2^k$	k	k	$(\{0,1\}^k)^{\leq L}$
◇ $\mathsf{poly}^{\mathcal{P}}$	$L/2^k$	$2/2^k$	k	$3\log(L)+\log(k)$	$(\{0,1\}^k)^{\leq L}$
Cascade (Merkle-Damgård) based on function $func : \{0,1\}^{k+b} \to \{0,1\}^k$					
MD	$L/2^k$		key for $func$	–	$\{0,1\}^{bL}$
MDR	$(L+1)/2^k$	$2/2^k$	key for $func$	$\log(L)$	$\{0,1\}^{bL}$
$(\varepsilon_{forge}, \varepsilon_{mac})$-$\Delta$ universal hash functions from ε-Δ universal ones					
Generic: for each $H(.)$ as below there is a "small" set $\mathcal{P}$ such that					
$H(.)$	$L^c/2^k$		key for H	–	$\{0,1\}^L$
◇ generic $H^{\mathcal{P}}(.)$	$(L+\ell)^c/2^k$	$2/2^k$	key for H	$\ell \approx (2c+1)\log(L)$	$\{0,1\}^L$

Fig. 1. Parameters for the hash functions considered in this paper. A leading ⋆ denotes a non-constructive, and a leading ◇ a non-uniform result. The bounds for the cascade construction assume that $func$ is a uniform random function, also the higher order terms that appear in the bounds for this construction are omitted in this table.

its security is mainly bounded by the "average-case" collision probability of the salted hash function, which can be much smaller than the "worst-case" probability which appears in the bound of the standard hash then encrypt approach. We now define what we mean by (the security of) a randomized MAC. Let us note that the definition given below becomes the standard definition for (deterministic) MACs when $\mathcal{R}$ is a singleton set.

Definition 2 (Randomized MAC) *A randomized message authentication scheme* MAC *is a function $\mathcal{S} \times \mathcal{X} \times \mathcal{P} \to \mathcal{Y}$. $\mathcal{P}$ is the randomness-space, $\mathcal{S}$ is the (secret) key-space, $\mathcal{X}$ the message domain and $\mathcal{Y}$ the tag-space.*

We denote with $\mathsf{FRG}_{\mathsf{MAC}}(q_{mac}, q_{forge})$ *the advantage of any adversary A in finding an existential forgery for* MAC *where A is allowed to ask for at most q_{mac} MACs and make q_{forge} forgery attempts. More formally we consider the following experiment: first $s \leftarrow \mathcal{S}$ is sampled, then A may query the MACing oracle, which*

$$\texttt{on input } m \texttt{ outputs } (\mathsf{MAC}(s,m,p),p) \quad \texttt{where} \quad p \leftarrow \mathcal{P}$$

at most q_{mac} times and a verification oracle which

on input (m,a,p) *outputs* 1 *if* $\mathsf{MAC}(s,m,p)=a$ *and* 0 *otherwise.*

at most q_{forge} times. Now $\mathsf{FRG}_{\mathsf{MAC}}(q_{mac},q_{forge})$ is an upper bound on the probability that any A succeeds in receiving 1 from the verification oracle on an input (m,a,p) where he did not already receive the output (a,p) on input m from the MACing oracle. Note that the adversary may choose the salt p when querying the verification oracle, but the MACing oracle chooses the p at random.

This definition is an information theoretic one as we did only bound the number of queries A is allowed to make but we make no other computational assumption. We can do this as we will consider only MACs which as a final step involve an application of a uniform random permutation. In reality one would have to replace this uniform random permutation (URP) with a pseudorandom permutation (PRP), as otherwise the construction is not practical, and to restrict the above definition to computationally bounded adversaries (as unbounded adversaries can distinguish a PRP from a URP). The security of such a computational MAC then can be upper bounded by $\mathsf{FRG}_{\mathsf{MAC}}(q_{mac},q_{forge})+\mathsf{Adv}_{\mathsf{PRP}}$ where $\mathsf{Adv}_{\mathsf{PRP}}$ is the distinguishing advantage for the pseudorandom permutation of the adversary considered. From now on, we will no longer mention this simple fact. The following proposition is well known.

Proposition 1 (Security of hash then encrypt) *Let $H : \mathcal{S}\times\mathcal{X}\rightarrow\mathcal{Y}$ be ε-universal and $f(.)$ a uniform random permutation over $\mathcal{Y}$.*

If $1/(|\mathcal{Y}|-q_{mac})\le\varepsilon$, then the MAC scheme with secret key $s\leftarrow\mathcal{S}$ where the authentication tag for a message $m\in\mathcal{X}$ is computed as

$$\mathsf{MAC}(s,m)=f(h_s(m))$$

has security

$$\mathsf{FRG}_{\mathsf{MAC}}(q_{mac},q_{forge})\le\varepsilon\cdot q_{mac}^2+\varepsilon\cdot q_{forge}$$

We do not prove this proposition, as it is just a special case of Theorem 1 below.

We now define the concept of salted hash functions described in the introduction.

Definition 3 ($(\varepsilon_{forge},\varepsilon_{mac})$-almost 2-universal salted hash function) *A hash function $H:\mathcal{S}\times\mathcal{X}\times\mathcal{P}\rightarrow\mathcal{Y}$ is $(\varepsilon_{forge},\varepsilon_{mac})$-universal if (below $x_1,x_2\in\mathcal{X}$, $p_1,p_2\in\mathcal{P}$ and $h_s(.,.)\overset{def}{=}H(s,.,.)$)*

$$\begin{aligned}\varepsilon_{forge}&\ge\max_{(x_1,p_1)\neq(x_2,p_2)}\mathsf{Pr}[s\leftarrow\mathcal{S};h_s(x_1,p_1)=h_s(x_2,p_2)]\\ \varepsilon_{mac}&\ge\max_{x_1,x_2,p_1}\mathsf{Pr}[s\leftarrow\mathcal{S};p_2\leftarrow\mathcal{P};h_s(x_1,p_1)=h_s(x_2,p_2)\wedge(x_1,p_1)\neq(x_2,p_2)]\end{aligned}$$

Every ε-universal hash function is an $(\varepsilon,\varepsilon)$-universal salted hash function with $\mathcal{P}=\emptyset$. We can now generalize the hash then encrypt paradigm to salted hash functions.

Theorem 1 (Security of salted hash then encrypt) *Let $H : \mathcal{S} \times \mathcal{X} \times \mathcal{P} \rightarrow \mathcal{Y}$ be $(\varepsilon_{forge}, \varepsilon_{mac})$-universal and $f(.)$ a uniform random permutation over $\mathcal{Y}$.*

If $1/(|\mathcal{Y}| - q_{mac}) \leq \varepsilon_{forge}$,[7] *then the MAC scheme with secret key $s \leftarrow \mathcal{S}$ where the authentication tag for a message $m \in \mathcal{X}$ is computed by first sampling $p \leftarrow \mathcal{P}$ and then setting*

$$\mathsf{MAC}(s, m, p) = (f(h_s(m, p)), p)$$

has security

$$\mathsf{FRG}_{\mathsf{MAC}}(q_{mac}, q_{forge}) \leq \varepsilon_{mac} \cdot q_{mac}^2 + \varepsilon_{forge} \cdot q_{forge}$$

Proof. Instead of bounding $\mathsf{FRG}_{\mathsf{MAC}}(q_{mac}, q_{forge})$ we bound the (larger) probability P that any adversary A can forge a MAC or he finds a collision. By a collision we mean that two outputs (a_1, p_1),(a_2, p_2) from the MACing oracle on two (not necessarily distinct) queries m_1 and m_2, satisfy $a_1 = a_2$ and $(m_1, p_1) \neq (m_2, p_2)$.

Let P_{col} denote the probability that a collision occurs before A found a forgery, and P_{frg} be the probability that A found a forgery before any collision occurred, so $P = P_{col} + P_{frg}$. Below we bound $P_{col} \leq \varepsilon_{mac} \cdot q_{mac}^2$ and $P_{frg} \leq \varepsilon_{forge} \cdot q_{forge}$ which then proves the theorem.

We can bound $P_{col} \leq \varepsilon_{mac} \cdot q_{mac}^2$ as follows: first, we can assume that A makes no forgery attempts (as trying to forge can only lower the probability of finding a collision *before* there was a successful forgery). Now we will show that for any $1 \leq i < j \leq q_{mac}$, the probability that the *first* collision is amongst the i'th and j'th query is at most ε_{mac}: as we are interested in the first collision, we can assume that $i = 1$ and $j = 2$, as making any intermediate queries can only lower the success probability (to see this, note that because of the application of the uniform random permutation, all the adversary learns about the outputs of the hash function is whether there were collisions or not). Next, for an adversary which makes only two queries, ε_{mac} is a trivial upper bound on the collision probability (even if we allow A to choose the salt for the first query). Applying the union bound we get that the probability that there are any $i, j, 1 \leq i < j \leq q_{mac}$ such that the i'th and j'th output collide, is at most $\varepsilon_{mac} \cdot q_{mac} \cdot (q_{mac} - 1)/2$, which thus is an upper bound for P_{col}.

We will now prove the bound $P_{frg} \leq \varepsilon_{forge} \cdot q_{forge}$. For any $j, 1 \leq j \leq q_{forge}$, let P_j denote the probability that the j'th forgery query is the first successful forgery and there was no collision before this forgery attempt. We will show $P_j \leq \varepsilon_{forge}$, this proves $P_{frg} \leq \varepsilon_{forge} \cdot q_{forge}$ as $P_{frg} = \sum_{j=1}^{q_{forge}} P_j$. To upper bound P_j we can assume that A skips the $j - 1$ first forgery attempts (this can only increase the success probability of the considered forgery attempt to be the *first* successful forgery). Moreover we allow A to chose all the salts for his (up

[7] This is satisfied if the hash function is input shrinking and ε_{forge} is at least slightly bigger than the optimal $1/|\mathcal{Y}|$, which is the setting that interests us. To see that the restriction is necessary, consider a hash function which is a permutation on $\mathcal{X} \times \mathcal{P} \equiv \mathcal{Y}$, then $\varepsilon_{forge} = \varepsilon_{mac} = 0$, but the forgery probability is not 0 as a random guess will always be successful with prob. $1/|\mathcal{Y}|$.

to q_{mac}) MACing queries he can ask before his forgery attempt. Again, this can only increase his success probability.

So we must upper bound (by ε_{forge}) the probability of any A winning the following game: A gets $\sigma_i = f(h_s(x_i))$ for $i = 1, \dots, k$ (with $k \leq q_{mac}$) and x_i's of his choice. Then he must come up with a σ, x (where $x \neq x_i$ for all $1 \leq i \leq k$). He wins if $f(h_s(x)) = \sigma$ and $\sigma_i \neq \sigma_j$ for all $1 \leq i < j \leq k$.

If A chooses a σ where $\sigma \neq \sigma_i$ for all $i = 1, \dots, k$, then the success probability (even conditioned on all σ_i being distinct), is at most $1/(|\mathcal{Y}| - k) \leq 1/(|\mathcal{Y}| - q_{mac}) \leq \varepsilon_{forge}$ (the last step by assumption), as then $\Pr[f(h_s(x)) = \sigma] \leq \Pr[f(h_s(x)) = \sigma | \forall i, 1 \leq i \leq k : h_s(x_i) \neq h_s(x)] = 1/(|\mathcal{Y}| - k)$, here the first step used that if $h_s(x_i) = h_s(x)$, then $\sigma = \sigma_i$, and the second used that f is a uniform random permutation and thus if $h_s(x)$ is new, then $f(h_s(x))$ is uniformly random over $\mathcal{Y} \setminus \{\sigma_i, \dots, \sigma_k\}$.

Now consider the other case, i.e. when A chooses a σ where for some i : $\sigma = \sigma_i$. From this A we construct an adversary A' which has at least the same probability of winning as follows. A' runs A and answers each of A's queries $x_1, \dots, x_k$ with uniformly random but distinct $\sigma'_1, \dots, \sigma'_k$. When now A outputs his forgery attempt $(\sigma = \sigma'_i, x)$, A' makes the single MACing query x_i, gets σ_i and outputs the forgery attempt $(\sigma = \sigma_i, x)$. It's not hard to verify that A' success probability is at least the one of A (it can be larger as A' will never lose the game due to collisions amongst the $\sigma_1, \dots, \sigma_k$, as he only asks one for each σ_i). Moreover A''s success probability is at most ε_{forge} as A' just chooses two values x, x_i before even using any oracle, and then wins if $h_s(x) = h_s(x_i)$. □

4 A Generic Construction

In this section we show that for every balanced ε-universal hash function H, where ε can even grow polynomially in the message length L, there exists a preprocessing using only $O(\log L)$ random bits, which makes the hash function $(\varepsilon_{forge}, \varepsilon_{mac})$-universal where $\varepsilon_{forge} \approx \varepsilon$ and ε_{mac} is of the smallest possible order.

Let $H : \mathcal{S} \times \{0,1\}^* \to \{0,1\}^k$ be $\varepsilon(.)$-almost 2-universal, by this we mean that for all $x_1, x_2 \in \{0,1\}^*$ where $x_1 \neq x_2, \ell = \max\{|x_1|, |x_2|\}$ and $h_s(.) \overset{\text{def}}{=} H(s, .)$

$$\mathsf{Pr}[s \leftarrow \mathcal{S}; h_s(x_1) = h_s(x_2)] \leq \varepsilon(\ell)$$

Definition 4 *A hash function H as above is balanced if for all $\ell \geq k$, $s \in \mathcal{S}$ and $y \in \{0,1\}^k$*

$$\mathsf{Pr}[x \leftarrow \{0,1\}^\ell; h_s(x) = y] = 2^{-k}$$

The following lemma states that from any such hash function H which is $\varepsilon(.)$-almost 2-universal and balanced we can get (non-constructively) a $(\varepsilon_{forge}, \varepsilon_{mac})$ salted hash function (with domain $\{0,1\}^L$ for any $L > k$) where $\varepsilon_{mac} = O(2/2^k)$ and $\varepsilon_{forge} = \varepsilon(L + r)$. Here r is the length of the public salt and if $\varepsilon(L) = O(L^c/2^k)$ we will get $r \approx (2c + 1)\log(L)$.

The construction is very simple, the salted hash function with key $s \in \mathcal{S}$, salt $p \in \{0,1\}^r$ and message x is computed as $h_s(g(p\|x))$ for some permutation g (we show that a random permutation is appropriate with high probability).

Lemma 1 *Let $H : \mathcal{S} \times \{0,1\}^* \to \{0,1\}^k$ be a balanced $\varepsilon(.)$-almost 2-universal hash function. Fix some integer $L \geq k$ and let r be the smallest integer satisfying*

$$2^r \geq \frac{2^{2k} \cdot \varepsilon(L+r)^2 \cdot (2L+r)}{\log(e)} \tag{1}$$

then there exists a permutation g over $\{0,1\}^{L+r}$ s.t. the salted hash function $H' : \mathcal{S} \times \{0,1\}^L \times \mathcal{P} \to \{0,1\}^k$ with $\mathcal{P} = \{0,1\}^r$ defined as

$$H'(s,m,p) \stackrel{def}{=} H(s, g(p\|m))$$

is $(\varepsilon_{forge}, \varepsilon_{mac})-$universal with $\quad \varepsilon_{forge} = \varepsilon(L+r) \qquad \varepsilon_{mac} = 2/2^k$.

Proof. Let $R \stackrel{\text{def}}{=} 2^r$ and $g_i(m) \stackrel{\text{def}}{=} g(i\|m)$. The bound on ε_{forge} is straightforward:

$$\begin{aligned}\varepsilon_{forge} &= \max_{x_1,x_2 \in \{0,1\}^L, p_1,p_2 \in \{0,1\}^r, (p_1,x_1) \neq (p_2,x_2)} \mathsf{Pr}[s \leftarrow \mathcal{S}; h_s(g_{p_1}(x_1)) = h_s(g_{p_2}(x_2))] \\ &= \max_{y_1,y_2 \in \{0,1\}^{L+r}, y_1 \neq y_2} \mathsf{Pr}[s \leftarrow \mathcal{S}; h_s(y_1) = h_s(y_2)] \\ &= \varepsilon(L+r)\end{aligned}$$

Above we used that $(p_1,x_1) \neq (p_2,x_2)$ implies $y_1 \neq y_2$ which holds as $y_1 = g(p_1\|x_1)$ and $y_2 = g(p_2\|x_2)$ and g is a permutation.

The proof for ε_{mac} is by the probabilistic method. We will show that a permutation g chosen at random has the desired property with probability > 0. For any $a, b \in \{0,1\}^L$ and $i, j \in \{0,1\}^r$ let $C_{i,j,a,b}$ denote the random variable (the probability is over g)

$$C_{i,j,a,b} = \mathsf{Pr}[s \leftarrow \mathcal{S}; h_s(g_i(a)) = h_s(g_j(b))]$$

As h_s is balanced we have for any $(i,a) \neq (j,b) : \mathsf{E}[C_{i,j,a,b}] \leq 1/2^k$, and as h_s is $\varepsilon(.)$-almost 2-universal and $|g_i(a)| = |g_j(b)| = L + r$

$$C_{i,j,a,b} \leq \varepsilon(L+r)$$

Let

$$C_{i,a,b} = \sum_{j \in \{0,1\}^r, (i,a) \neq (j,b)} C_{i,j,a,b} \tag{2}$$

As the sum ranges over R terms (resp. $R-1$ if $a = b$) we have $\mathsf{E}[C_{i,a,b}] \leq R/2^k$. We can apply Hoeffding's inequality (see Appendix A), for the case $a \neq b$ (then the sum in eq.(2) has exactly R terms, if $a = b$ then we have only $R-1$ terms and can use the same bound as proven below) we get

$$\mathsf{Pr}\left[C_{i,a,b} \geq 2 \cdot \mathsf{E}[C_{i,a,b}]\right] < \exp\left(-\frac{2 \cdot (R/2^k)^2}{R \cdot \varepsilon(L+r)^2}\right) \leq 2^{-2L-r}$$

Where in the last step we used (1) (recall that $R \overset{\text{def}}{=} 2^r$). So there is a g such that $C_{i,a,b} \leq 2 \cdot \mathsf{E}[C_{i,a,b}] \leq 2R/2^k$ for all $i \in \{0,1\}^r$ and $a, b \in \{0,1\}^L$, for this g

$$\begin{aligned}
\varepsilon_{mac} &= \max_{\substack{i\in\{0,1\}^r\\ a,b\in\{0,1\}^L}} \mathsf{Pr}[j \leftarrow \{0,1\}^r; s \leftarrow \mathcal{S}; h_s(g_i(a)) = h_s(g_j(b)) \wedge ((i,a) \neq (j,b))] \\
&= \max_{i\in\{0,1\}^r, a,b\in\{0,1\}^L} R^{-1} \sum_{j\in\{0,1\}^r,(i,a)\neq(j,b)} C_{i,j,a,b} \\
&= R^{-1} \max_{i\in\{0,1\}^r, a,b\in\{0,1\}^L} C_{i,a,b} \\
&\leq 2^{-k+1}
\end{aligned}$$

□

5 poly: Hashing by Polynomial Evaluation

A popular way of ε-almost universal hashing is to parse the message into coefficients of a polynomial over some field (we will use $GF(2^k)$) and evaluate it on a random point. We propose a simple randomized preprocessing for this hash function: just set the constant coefficient at random. If this coefficient is set uniformly at random, this gives a salted hash function with an optimal $\varepsilon_{mac} = 1/2^k$. We then show that one can also sample the coefficient from a small set, thus using fewer randomness, and still achieve an almost optimal $\varepsilon_{mac} \leq 2/2^k$. We will come back to this construction later in Section 7, where we prove some generic results which imply Lemma 3 and Lemma 4 from this section (though with somewhat worse parameters).

Definition 5 *For $M = (M_1, \ldots, M_m)$ (each $M_i \in GF(2^k) \cong B_k$) we denote with $f_M(.)$ the polynomial of degree $m-1$ over $GF(2^k)$ given by*

$$f_M(x) = \sum_{i=1}^{m} M_i \cdot x^{i-1}$$

Definition 6 *With* poly *we denote the hash function which on input $M \in GF(2^k)^* \cong B_k^*$ with key $s \leftarrow GF(2^k)$ is computed as* $\mathsf{poly}_s(M) = f_M(s)$.

Lemma 2 (see [22]) poly *with domain $B_k^{\leq L}$ is ε-almost universal with $\varepsilon = (L-1)/2^k$.*

Definition 7 $\mathsf{poly}^{GF(2^k)}$ *is the salted hash function with secret key part $s \leftarrow GF(2^k)$ and public salt $p \leftarrow GF(2^k)$ which on input $M \in B_k^*$ is computed as*

$$\mathsf{poly}^{GF(2^k)}_{s,p}(M) = f_{(p,M)}(s)$$

Lemma 3 $\mathsf{poly}^{GF(2^k)}$ *with domain $B_k^{\leq L}$ is $(\varepsilon_{forge}, \varepsilon_{mac})$-universal where*

$$\varepsilon_{forge} = L/2^k \tag{3}$$
$$\varepsilon_{mac} = 1/2^k \tag{4}$$

The bound on ε_{forge} follows from Lemma 2, and the bound on ε_{mac} is obvious. We now consider another salted version of the poly hash function which is similar to $\mathsf{poly}^{GF(2^k)}$ but where the public salt is not chosen from the whole of $GF(2^k)$ but only from a subset $\mathcal{P} \subset GF(2^k)$.

Definition 8 *For any $\mathcal{P} \subset GF(2^k)$ we denote with $\mathsf{poly}^{\mathcal{P}}$ the salted hash function with secret key part $s \leftarrow GF(2^k)$ and public salt $p \leftarrow \mathcal{P}$ which on input $M \in GF(2^k)^* \cong B_k^*$ is computed as*

$$\mathsf{poly}^{\mathcal{P}}_{s,p}(M) = f_{(p,M)}(s)$$

We will show (constructively, but "non-uniformly") that there is a "small" $\mathcal{P}$ such that the construction is $(\varepsilon_{mac}, \varepsilon_{forge})$-universal with $\varepsilon_{mac} = 2/2^k$. Namely, a random "small" $\mathcal{P}$ works with all but negligible probability (in particular, once such $\mathcal{P}$ is chosen *once*, it can be fixed *forever* and "hardwired" into the implementation).

Lemma 4 *For any $L \in \mathbb{N}$ and a random subset $\mathcal{P} \subset GF(2^k)$ of size $|\mathcal{P}| = \frac{k(L+2)L^2}{\log(e)}$, with probability $1-2^{-k}$ (over the choice of $\mathcal{P}$) the hash function $\mathsf{poly}^{\mathcal{P}}$ with domain $B_k^{\leq L}$ is $(\varepsilon_{forge}, \varepsilon_{mac})$-universal with*

$$\varepsilon_{forge} = L/2^k \tag{5}$$

$$\varepsilon_{mac} = 2/2^k \tag{6}$$

Proof. The bound (5) on ε_{forge} follows from Lemma 2 (and holds for any $\mathcal{P}$).

To prove the bound (6) on ε_{mac} we must show that a $\mathcal{P}$ chosen at random has the claimed property with probability $1-2^{-k}$. For a polynomial f over $GF(2^k)$ we denote with $z(f) = |\{x \in GF(2^k) : f(x) = 0\}|$ the number of zeros of f. Let f be any polynomial over $GF(2^k)$ of degree at most L and (for some m to be defined) let $\mathcal{P} = (p_1, \ldots, p_m)$ denote a subset of $GF(2^k)$ sampled uniformly at random (with repetition).

X_i denotes the random variable $z(f_i)$ where f_i is $f + p_i$ (i.e. f with p_i added to the constant coefficient). As p_i is random we have $\mathsf{Pr}[f_i(x) = 0] = 1/2^k$ for any $x \in GF(2^k)$ and thus

$$\mathsf{E}[X_i] = \sum_{x \in GF(2^k)} \mathsf{Pr}[f_i(x) = 0] = 1$$

and as any polynomial of degree L has at most L roots

$$0 \leq X_i \leq L$$

Let $S = X_1 + X_2 + \ldots + X_m$, we have $E[S] = m$ and by the Hoeffding bound [12]

$$\mathsf{Pr}[S - E[S] \geq m] = \mathsf{Pr}[S \geq 2m] \leq \exp\left(-\frac{m^2}{m \cdot L^2}\right) \tag{7}$$

which for $m = \frac{k(L+2)L^2}{\log(e)}$ is less than $2^{-k(L+2)}$. Taking the union bound over all $2^{k(L+1)}$ polynomials of degree $\leq L$, we get the probability that (7) is not satisfied for at least one of them is at most $2^{-k(L+2)} \cdot 2^{k(L+1)} = 2^{-k}$.

To conclude the proof we must still show that any $\mathcal{P}$ which satisfies (7) for all polynomials of degree $\leq L$ also satisfies (6). ε_{mac} is the maximum over $M \in GF(2^k)^{L+1}$ (with the first element from $\mathcal{P}$, but we will not use that) and $M' \in GF(2^k)^L$ of

$$\varepsilon_{mac} \geq \mathsf{Pr}_{s \leftarrow GF(2^k), p \leftarrow \mathcal{P}}[f_M(s) = f_{(p,M')}(s)]$$

Which for $f = f_{(0^k,M')} - f_M$ and $f_i = f + p_i$ we can write as

$$\mathsf{Pr}_{s \leftarrow GF(2^k), p \leftarrow \mathcal{P}}[f(s) + p = 0] = \sum_{i=1}^{m} \mathsf{Pr}[p = p_i] z(f_i)/2^k \leq 2/2^k$$

In the last step we used that we chose our $\mathcal{P}$ such that $\sum_{i=1}^{m} z(f_i) \leq 2m$ for all f and $\mathsf{Pr}[p = p_i] = 1/m$. □

6 Cascade Construction

In his section we consider the Merkle-Damgård construction. Here a preprocessing which simply appends a few random bits to the message gives a salted hash function with good parameters.

For a function $\xi : \{0,1\}^k \times \{0,1\}^b \to \{0,1\}^k$ we denote with $\mathsf{MD}_\xi : B_b^* \to B_k$ the cascade (aka. Merkle-Damgård) construction based on ξ which on input $M = M_1 \| \ldots \| M_m$, each $M_i \in B_b$, outputs X_m which is recursively defined as $X_0 = 0^k$, and $X_i = \xi(X_{i-1}, M_i)$.

Definition 9 (MDR) *For $k, b, L \in \mathbb{N}$ where $b \geq \log(L)$ we denote with $\mathsf{MDR}^L : B_b^{L-1} \to B_k$ the salted hash function whose secret key part is a uniformly random function $\xi : \{0,1\}^k \times \{0,1\}^b \to \{0,1\}^k$ and the public salt is $r \leftarrow \{0,1\}^{\lceil \log(L) \rceil}$. The randomized hash value on input $M \in B_b^*$ is computed as*

$$\mathsf{MDR}^L(M) = \mathsf{MD}_\xi(M \| \langle r \rangle_b)$$

Lemma 5 *$\mathsf{MDR}^L : B_b^{L-1} \to B_k$ is $(\varepsilon_{forge}, \varepsilon_{mac})$-universal with*

$$\varepsilon_{mac} = \frac{2}{2^k} + O(L^3/2^{2k}) \tag{8}$$

$$\varepsilon_{forge} = \frac{L}{2^k} + O(L^3/2^{2k}) \tag{9}$$

Proof. The proof follows almost directly form Propositions 1 and 2 from [8]. Proposition 2 from [8] states that for a random ξ and any $M \neq M' \in B_b^L$

$$\mathsf{Pr}_\xi[\mathsf{MD}_\xi(M) = \mathsf{MD}_\xi(M')] \leq L/2^k + O(L^3/2^{2k})$$

which directly gives the bound (9) on ε_{forge}. Proposition 1 from [8] states that if M and M' differ in the last b-bit block, then an even better bound

$$\mathsf{Pr}_\xi[\mathsf{MD}_\xi(M) = \mathsf{MD}_\xi(M')] \leq 1/2^k + O(L^2/2^{2k})$$

applies. To bound ε_{mac} we can use that MDR adds a random last block $r \leftarrow \{0,1\}^{\lceil \log(L) \rceil}$ to the message, which then will be equivalent to the last block of the other message with prob. at most $\phi \leq 1/L$ and we get

$$\varepsilon_{mac} \leq (1-\phi) \cdot \frac{1}{2^k} + \phi \cdot \frac{L}{2^k} + O(L^3/2^{2k}) \leq \frac{2}{2^k} + O(L^3/2^{2k}).$$

□

7 A Generic Construction from ϵ-Δ Universal Hash Functions

In this section we consider hash functions which are not only ε-universal, but satisfy the stronger notion of ε-Δ universality.

Definition 10 (ε-Δ universal hash function [16]) *A hash function $H : \mathcal{S} \times \mathcal{X} \rightarrow \mathcal{Y}$, where $\mathcal{Y}$ is an additive Abelian group, is ε-Δ universal if for all $x, y \in \mathcal{X}$,$x \neq y$ and $c \in \mathcal{Y}$*

$$\mathsf{Pr}[s \leftarrow \mathcal{S}; h_s(x) - h_s(y) = c] \leq \varepsilon$$

It is easy to see (and stated as Proposition 2 below) that adding a value chosen uniformly at random to the output of a ε-Δ universal hash function gives a $(\varepsilon_{forge}, \varepsilon_{mac})$-universal hash function with $\varepsilon_{forge} = \varepsilon$ and an optimal $\varepsilon_{mac} = 1/|\mathcal{Y}|$.

The main result of this section is a theorem which states that for every ε-Δ universal hash function, there always exists a randomized *post*processing, which only uses a logarithmic number of random bits and makes the hash function $(\varepsilon_{forge}, \varepsilon_{mac})$-universal where $\varepsilon_{forge} = \varepsilon$ and ε_{mac} is close to optimal. By post-processing we mean that only the hash value of the message, but not the message itself, must be randomized.

Let us remark that the polynomial construction from Section 5 is $L/2^k$-Δ universal if the constant coefficient (i.e. the first message block) is fixed, say 0^k. With this observation Lemma 3 follows directly from Proposition 2, and Lemma 4 (with somewhat worse parameters) follows from Theorem 2 we prove below.

Definition 11 ($(\varepsilon_{forge}, \varepsilon_{mac})$-$\Delta$ universal hash function)
For an ε-Δ universal hash function $H : \mathcal{S} \times \mathcal{X} \rightarrow \mathcal{Y}$ and a set $\mathcal{P} \subseteq \mathcal{Y}$ we say that $H^{\mathcal{P}}$ is $(\varepsilon_{forge}, \varepsilon_{mac})$-$\Delta$ universal if for any $p_1, p_2 \in \mathcal{P}, x_1, x_2 \in \mathcal{X}$ where $(p_1, x_1) \neq (p_2, x_2)$

$$\varepsilon_{forge} \geq \mathsf{Pr}[s \leftarrow \mathcal{S}; h_s(x_1) + p_1 = h_s(x_2) + p_2]$$

and

$$\varepsilon_{mac} \geq \mathsf{Pr}[s \leftarrow \mathcal{S}; p \leftarrow \mathcal{P}; h_s(x_1) + p_1 = h_s(x_2) + p \land (p_1, x_1) \neq (p, x_2)]$$

Proposition 2 *If $H : \mathcal{S} \times \mathcal{X} \to \mathcal{Y}$ is ε-Δ universal, then $H^{\mathcal{Y}}$ is $(\varepsilon, 1/|\mathcal{Y}|)$-$\Delta$ universal.*

Theorem 2 *If $H : \mathcal{S} \times \mathcal{X} \to \mathcal{Y}$ is ε-Δ universal, then there exists a $\mathcal{P} \subset \mathcal{Y}$ of size $m = |\mathcal{P}|$ such that*

$$m \leq \ln(|\mathcal{X}|^2 \cdot m) \cdot |\mathcal{Y}|^2 \cdot \varepsilon^2 \tag{10}$$

and $H^{\mathcal{P}}$ is $(\varepsilon, 2/|\mathcal{Y}|)$-$\Delta$ universal.

Proof. The proof is by the probabilistic method. We show that a random subset $\mathcal{P} = \{p_1, \ldots, p_m\}$ of $\mathcal{Y}$ of size m which satisfies (10) has the claimed property with probability > 0 and thus exists.

For $i, j : 1 \leq i, j \leq m$ and $a, b \in \mathcal{X}$ where $(i, a) \neq (j, b)$ let let $C_{i,j,a,b}$ denote the random variable (the probability is over the choice of $\mathcal{P}$)

$$C_{i,j,a,b} = \mathsf{Pr}[s \leftarrow \mathcal{S}; h(a) + p_i = h(b) + p_j]$$

and for $(i, a) = (j, b)$ we set $C_{i,j,a,b} = 0$. Clearly for $(i, a) \neq (j, b)$

$$\mathsf{E}[C_{i,j,a,b}] = 1/|\mathcal{Y}|$$

Now consider the random variable

$$C_{i,j,b} = \sum_{j \in \mathcal{X}} C_{i,j,a,b}$$

We have $\mathsf{E}[C_{i,a,b}] \leq m/|\mathcal{Y}|$ and for $a \neq b$ we get by the Hoeffding bound (see Appendix A) an upper bound for the probability that $C_{i,a,b}$ is more that twice its expected value (for $a = b$ the bound is even slightly better)

$$\begin{aligned}
\mathsf{Pr}\left[C_{i,a,b} \geq 2 \cdot \mathsf{E}[C_{i,a,b}]\right] &< \exp\left(-\frac{2 \cdot (m/|\mathcal{Y}|)^2}{m \cdot \varepsilon^2}\right) \\
&\leq \exp\left(-\frac{m}{|\mathcal{Y}|^2 \cdot \varepsilon^2}\right) \\
&\leq \frac{1}{|\mathcal{X}|^2 \cdot m}
\end{aligned}$$

So there is a $\mathcal{P}$ such that $C_{i,a,b} \leq 2 \cdot \mathsf{E}[C_{i,a,b}] \leq 2 \cdot m/|\mathcal{Y}|$ is satisfied for all $i \in [m]$ and $a, b \in \mathcal{X}$. For this $\mathcal{P}$ we get

$$\begin{aligned}
\varepsilon_{mac} &= \max_{i \in [m], a, b \in \mathcal{X}} \mathsf{Pr}[j \leftarrow [m]; s \leftarrow \mathcal{S}; h_s(a) + p_i = h(b) + p_j) \wedge ((p_i, a) \neq (p_j, b))] \\
&= m^{-1} \max_{i \in [m], a, b \in \mathcal{X}} \sum_{j \in [m]} C_{i,j,a,b} \\
&= m^{-1} \max_{i \in [m], a, b \in \mathcal{X}} C_{i,a,b} \\
&\leq 2/|\mathcal{Y}|
\end{aligned}$$

□

To get an intuition what eq. (10) means, assume we start with a hash function which maps L-bit strings to k-bit strings and which is $L^c/2^k$-Δ universal for some $c > 0$, so $|\mathcal{X}| = 2^L$ and $|\mathcal{Y}| = 2^k$. Now (10) means

$$m \leq \frac{(2 \cdot L + \log m) \cdot L^{2c}}{\log e}$$

or assuming $\log(m) \leq L$

$$\log(m) \leq \log 3 - \log e + (2c+1)\log L < 2 + (2c+1)\log L \tag{11}$$

The assumption $\log(m) \leq L$ holds for all $L \geq 2 + (2c+1)\log L$, thus for such L also (11) is satisfied. So to sample from $\mathcal{P}$ we need $O(1) + (2c+1)\log L$ random bits.

8 Numerical Example

In this section we give a numerical example to illustrate how the classical upper bound $\varepsilon \cdot q_{mac}^2 + \varepsilon \cdot q_{forge}$ for the forging probability from Proposition 1 compares to the $\varepsilon_{mac} \cdot q_{mac}^2 + \varepsilon_{forge} \cdot q_{forge}$ upper bound (for salted hash then encrypt) given in Theorem 1. We will use polynomial hashing poly and its salted version $\mathsf{poly}^{\mathcal{P}}$ as considered in Section §5. We take a (standard) tag-length of $k := 128$ bits and the messages to be signed are of size 128MB (so each message has $L := 2^{23}$ blocks of size 128 bits). Now consider an adversary which can request $q_{mac} := 2^{40}$ MACs and make up to $q_{forge} := 2^{40}$ forgery attempts. For these parameters, poly is ε-universal with $\epsilon = (L-1)/2^{128} \approx 2^{-105}$, and the bound from Proposition 1 gives an upper bound on the forging probability for poly of

$$\varepsilon \cdot q_{mac}^2 + \varepsilon \cdot q_{forge} \approx \frac{2^{80}}{2^{105}} + \frac{2^{40}}{2^{105}} \approx 2^{-25}. \tag{12}$$

For the salted version $\mathsf{poly}^{\mathcal{P}}$, which as shown in Lemma 4 has $\varepsilon_{forge} = L/2^{128} = 2^{-105}$ and $\varepsilon_{mac} = 2^{-127}$, we get with Theorem 1 an upper bound of

$$\varepsilon_{mac} \cdot q_{mac}^2 + \varepsilon_{forge} \cdot q_{forge} = \frac{2^{80}}{2^{127}} + \frac{2^{40}}{2^{105}} \approx 2^{-47}. \tag{13}$$

So, compared to the classical bound (12), the security guarantee is better by a factor of $\approx 2^{22} \approx 4 \cdot 10^6$. Note that this gap is basically $L/2$, this will always be the case for (the most interesting) range of parameters where $\varepsilon_{mac} \cdot q_{mac}^2 \gg \varepsilon_{forge} \cdot q_{forge}$. The length of the random salt used in this construction (with these parameters) is $3\log(L) + \log(k) = 3 \cdot 23 + 7 = 76$ bits.

For the cascade construction MD and its salted version MDR as considered in Section §6 (with $b = k = 128$), we get the same bounds (12) and (13) respectively,[8] but the length of the random salt needed for this construction is considerably shorter, $\log(L) = 23$ bits compared to the 76 bits needed for $\mathsf{poly}^{\mathcal{P}}$.

[8] The reason is that the bounds on $\varepsilon_{forge}, \varepsilon_{mac}$ for MDR as given in Lemma 5 are identical to the bounds for $\mathsf{poly}^{\mathcal{P}}$ from Lemma 4 up to a $O(L^3/2^k)$ term, which for the parameters considered here will be irrelevant.

9 Related Work and Open Problems

Following the initial papers of Carter and Wegman [7, 26], foundations of universal hash function-based authentication were laid by [23, 16, 19, 24].

The analysis of the folklore polynomial construction is well known (see [5] for some history). The Merkle-Damgård functions was analyzed as an ε-universal hash function by [2, 8], the latter proving a bound of $\varepsilon \approx L/2^k$ for hashing an Lb-block input using the compression function $\xi : \{0,1\}^k \times \{0,1\}^b \to \{0,1\}^k$ (modeled as a truly random function; here $L < 2^{k/2}$).

It is also interesting to compare the "hash then encrypt" approach we study here with a variant of this approach studied by [26, 16], which actually led to the introduction of ε-Δ universal hash functions. Here one replaces a block cipher by a "fresh one-time pad". In modern terminology, the MAC has a tag of the form $(r, h_s(m) \oplus f_t(r))$, where f_t is a pseudorandom function with a new key t, and r is a fresh "nonce" which is not supposed to repeat. In practice, this means that r is either a counter, or a fresh random value. In the first case, we get *perfect* security (aside from the insecurity of f).[9] However, maintaining a counter introduces state, and stateful MACs are extremely inconvenient in many situations (see [21, 15] for several good reasons). Correspondingly, to make a fair comparison we should only consider the case when r is a fresh random salt (in which case the MAC is indeed stateless, but r has to be part of the tag). To make such a comparison, let us fix the output of the hash function h_s to $\{0,1\}^k$, and replace the PRF by a truly random function. Then, we get that the length of the tag in the "XOR-scheme" is $|r|+k$, while its exact security is $\varepsilon + O(q^2/2^{|r|})$ (where the last term comes from the birthday bound measuring collisions on r). Thus, to get to the desired overall security of $O(q^2/2^k)$, this randomized MAC must use $|r| = \Omega(k)$ random bits and increase the tag length by this amount as well. On the other hand, we demonstrated that our randomized MACs can achieve the same level of security using only $|p| = O(\log L)$ bits of randomness, which is asymptotically smaller than k. In other words, although using the one-time pad has other advantages over using the block cipher,[10] it is *provably inferior* to our method for achieving $O(q^2/2^k)$ security (via *stateless* MACs).

We also mention that message preprocessing has been used previously in other contexts. For example, we already mentioned the work of Jaulmes et al. [15] on RMAC. Semanko [21] investigated the security of iterated MACs (like the cascade or CBC construction), which are randomized by prepending a random string. Here finding collisions does not necessarily imply a new forgery, but Semanko showed some non-obvious forgery attacks. In particular, even when prepending up to $k/2$ random bits one can find a forgery after $2^{k/2}$ queries (and not just a collision, which by the birthday bound is trivial). Let us stress that in

[9] Bernstein [4] investigates how much security one loses when f is a *permutation* (like AES).

[10] E.g., it can go below the $O(q^2/2^k)$ barrier, even when the hash function output is fixed to k.

our case, the length of random salt is small enough so that the above distinction between collisions and forgeries does not play any significant role.

Bellare et al. [2] also used randomness to improve the security of the cascade construction for a (stronger than ours) task of domain extension of a pseudorandom function (in which case there is no need to add the encryption step at the end). In particular, the message preprocessing used there is different from ours (prepend instead of append), and the exact security is weaker as well. The same paper also considers changing the cascade construction by appending a fixed but random secret string t to each message (instead of choosing a fresh public t per each message). However, this is done for a different purpose of achieving "prefix-freeness" of messages.

Recently, Halevi and Krawczyk [9] also proposed to use randomized message preprocessing in the design of signature schemes. Their goal is to lower the *computational* assumptions on the hash function used in signature schemes. In particular, they show that randomized versions of some signature schemes based on hash functions only require second-preimage resistance while the original scheme needed (the stronger) collision resistance. This works continues and optimizes the more general direction of replacing a fixed collision-resistant hash function in the "hash then sign" paradigm by a universal one-way hash function [13] chosen at random for each new message signed (and appended to the message before signing). Notice, while the main goal of [9] is to reduce the needed computational assumption on the hash function (and not the length of the salt or the actual exact security), our setting is information-theoretic with the primary goal of improving the exact security while simultaneously minimizing the salt length.

Amortized Collision Probability. Another possibility to get better bounds on the exact security of the "hash then encrypt" paradigm is to consider "amortized" collision probability. For any q messages $x_1, \ldots, x_q$ and a ε-universal hash-function, the probability that $h(x_i) = h(x_j)$ for any $i \neq j$ is in $O(\varepsilon q^2)$. This $O(\varepsilon q^2)$ bound is proven by applying the union bound to all $\binom{q}{2}$ pairs of authenticated messages, which introduces some slackness: even if there are some pairs of messages with collision probability ε, it is not clear whether there exist $q \gg 2$ messages where each (or most) pairs collide with probability $\Omega(\varepsilon)$, and, moreover, the collision probabilities for distinct pairs are sufficiently independent.[11] Thus, it is possible that the actual collision probability is much less than $O(\varepsilon q^2)$. We know of one example where this has been proven to be the case, namely the CBC-function (based on uniformly random permutations). This function is ε-universal with $\varepsilon = \Theta(L^{1/\ln \ln L})$ [3], but using the amortized approach described above (assuming $q \geq L^2$ and $L \leq 2^{k/8}$) one can prove [18] an optimal $O(q^2/2^k)$ security bound, despite $\varepsilon = \omega(1/2^k)$.

Of course, this raises the question if the other constructions we consider also already achieve $O(q^2/2^k)$ security (for a non-trivial range of parameters)

[11] To see why those probabilities must be independent, consider an h where either all or none of the messages collide, then the collision probability for all q messages is only $O(\varepsilon)$ and not $O(\varepsilon q^2)$.

without randomization. As to the cascade construction, this is easily seen not to be the case, as there are q messages of length L where the collision probability is $\Theta(Lq^2/2^k)$. As to the polynomial construction, we do not know the answer, but conjecture that one cannot achieve the optimal security $O(q^2/2^k)$ (as we do with randomization).

References

1. Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In *CRYPTO*, pp. 1–15, 1996.
2. Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Pseudorandom functions revisited: The cascade construction and its concrete security. In *FOCS*, pp. 514–523, 1996.
3. Mihir Bellare, Krzysztof Pietrzak, and Phillip Rogaway. Improved Security Analyses for CBC MACs. In *Advances in Cryptology — CRYPTO '05*, August 2005.
4. Daniel J. Bernstein, Stronger Security Bounds for Wegman-Carter-Shoup Authenticators. In *EUROCRYPT*, pp. 164–180, 2005.
5. Daniel J. Bernstein, The Poly1305-AES Message-Authentication Code. In *FSE*, pp. 32–49, 2005.
6. Antoon Bosselaers and Bart Preneel, editors. *Integrity Primitives for Secure Information Systems, Final Report of RACE Integrity Primitives Evaluation RIPE-RACE 1040*, volume 1007 of *Lecture Notes in Computer Science*. Springer, 1995.
7. Larry Carter and Mark N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences (JCSS)*, 18:143–154, 1979.
8. Yevgeniy Dodis, Rosario Gennaro, Johan Håstad, Hugo Krawczyk, and Tal Rabin. Randomness Extraction and Key Derivation Using the CBC, Cascade and HMAC Modes. In *CRYPTO*, pages 494–510, 2004.
9. Shai Halevi and Hugo Krawczyk. Strengthening digital signatures via randomized hashing. In *CRYPTO*, 2006.
10. G. Hardy and E. Wright. *An Introduction to the Theory of Numbers.* Oxford University Press, 1980.
11. Tzvika Hartman and Ran Raz. On the distribution of the number of roots of polynomials and explicit weak designs. *Random Struct. Algorithms*, 23(3):235–263, 2003.
12. Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *J. Amer. Statist. Assoc.*, vol. 58:13–30, 1963.
13. Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *STOC*, pages 33–43, 1989.
14. Svante Janson, Tomasz Luczak, and Andrzej Rucinski. *Random Graphs.* Wiley, 2000.
15. Éliane Jaulmes and Antoine Joux and Frédéric Valette, On the Security of Randomized CBC-MAC Beyond the Birthday Paradox Limit: A New Construction. In *FSE*, pp. 237–251, 2002.
16. Hugo Krawczyk. *LFSR-Based Hashing and Authentication.* In *CRYPTO*, pp. 129–139, 1994.
17. Erez Petrank and Charles Rackoff. CBC MAC for real-time data sources. *Journal of Cryptology*, pages 315–338, 2000.
18. Krzysztof Pietrzak. A tight bound for EMAC. In *ICALP (2)*, pages 168–179, 2006.

19. Phillip Rogaway. Bucket Hashing and Its Application to Fast Message Authentication. In *CRYPTO*, pp. 29–42, 1995.
20. J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities.. *J. ACM*, 27(4):701–717, 1980.
21. Michael Semanko. L-collision Attacks against Randomized MACs. In *CRYPTO*, pp. 216–228, 2000.
22. Victor Shoup. *A Computational Introduction to Number Theory and Algebra.* Cambridge University Press, 2005.
23. Douglas R. Stinson. Universal Hashing and Authentication Codes. *Designs, Codes and Cryptography*, 4:369–380, 1994.
24. Douglas R. Stinson. On the connections between universal hashing, combinatorial designs and error-correcting codes. In *Congressus Numerantium* 114:7–27, 1996.
25. Douglas R. Stinson. Personal Communication, 2005.
26. Mark N. Wegman and Larry Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences (JCSS)*, 22(3):265–279, 1981.

A Hoeffding Bound

Let $X_1, \ldots, X_n$ be independent random variables. Further assume the X_i are bounded, i.e. for each $i = 1, \ldots, n$ there are a_i, b_i such that

$$\mathsf{Pr}[a_i \leq X_i \leq b_i] = 1$$

then for the sum $S = X_1 + \ldots X_n$ we have for any $t \geq 0$

$$\mathsf{Pr}[S - \mathsf{E}[S] \geq nt] \leq \exp\left(-\frac{2n^2t^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right)$$

We will only use the case where the X_i are identically distributed such that $0 \leq X_i \leq \varepsilon$ and are only interested in the probability that $S \geq 2\mathsf{E}[S]$, so if γ is an upper bound on $\mathsf{E}[S]$ we use

$$\mathsf{Pr}[S \geq 2 \cdot \mathsf{E}[S]] \;\leq\; \mathsf{Pr}[S - \mathsf{E}[S] \geq \gamma] \;\leq\; \exp\left(-\frac{2 \cdot \gamma}{n \cdot \varepsilon^2}\right) \tag{14}$$

In our applications the X_i's are not completely independent, but chosen at random from some finite set *without repetition.* Fortunately the Hoeffding bound also applies in this case as proven in section 6 of the original paper [12] (see also Theorem 2.10 of [14]).