Efficient Binary Conversion for Paillier Encrypted Values

Berry Schoenmakers[1] and Pim Tuyls[2]

[1] Dept. of Mathematics and Computing Science, TU Eindhoven,
P.O. Box 513, 5600 MB Eindhoven, The Netherlands.
berry@win.tue.nl
[2] Philips Research Labs
Prof. Holstlaan 4, 5656 AA Eindhoven, The Netherlands
pim.tuyls@philips.com

Abstract. We consider the framework of secure n-party computation based on threshold homomorphic cryptosystems as put forth by Cramer, Damgård, and Nielsen at Eurocrypt 2001. When used with Paillier's cryptosystem, this framework allows for efficient secure evaluation of any arithmetic circuit defined over $\mathbb{Z}_N$, where N is the RSA modulus of the underlying Paillier cryptosystem.

In this paper, we extend the scope of the framework by considering the problem of converting a given Paillier encryption of a value $x \in \mathbb{Z}_N$ into Paillier encryptions of the bits of x. We present solutions for the general case in which x can be any integer in $\{0, 1, \ldots, N-1\}$, and for the restricted case in which $x < N/(n2^\kappa)$ for a security parameter κ. In the latter case, we show how to extract the ℓ least significant bits of x (in encrypted form) in time proportional to ℓ, typically saving a factor of $\log_2 N/\ell$ compared to the general case.

Thus, intermediate computations that rely in an essential way on the binary representations of their input values can be handled without enforcing that the *entire* computation is done bitwise. Typical examples involve the relational operators such as $<$ and $=$. As a specific scenario we will consider the setting for (approximate) matching of biometric templates, given as bit strings.

1 Introduction

We consider secure n-party computation in the framework based on threshold homomorphic cryptosystems, as put forth by Cramer, Damgård, and Nielsen [CDN01]. To evaluate a given n-ary function f securely, one expresses f as an arithmetic circuit C composed of elementary gates, such as addition gates and multiplication gates. Given ciphertexts $[\![x_1]\!], \ldots, [\![x_n]\!]$, the gates are evaluated one by one, ultimately resulting in a ciphertext $[\![f(x_1, \ldots, x_n)]\!]$. Here, $[\![x]\!]$ denotes a probabilistic encryption of x in the underlying threshold homomorphic cryptosystem, which will be the Paillier cryptosystem [Pai99] throughout most of this paper. The homomorphic property ensures that evaluation of addition gates is essentially for free, as addition may simply be done by multiplying two ciphertexts. For multiplication gates, however, one needs a joint protocol involving at least one threshold decryption in the underlying cryptosystem.

An important feature of this type of protocols for secure computation is that the communication complexity, which is the dominating complexity measure, is $O(nk|C|)$ bits, where k is a security parameter, and $|C|$ is the number of (multiplication) gates of circuit C. Moreover, if Paillier is used as the underlying cryptosystem, the result of [CDN01] is particularly efficient in handling arithmetic with *large* numbers. Each arithmetic gate handles integer values modulo N, where N is the RSA modulus used for Paillier, at a cost *independent* of the size of these integers.

Arithmetic circuits for functions that also involve some bit-oriented steps tend to be inefficient, as can be seen from the following simple but somewhat contrived example. Consider the function $ws(x_1, \ldots, x_n)$ which outputs the Hamming weight of the binary representation of the sum $\sum_{i=1}^{n} x_i$ of the inputs x_i, which are assumed to be from a bounded range, $0 \leq x_i < 2^{64}$ say. Lacking an efficient secure conversion of an integer value into its binary presentation, the natural thing to do would be to require that the inputs x_i are given bitwise (hence as 64 encrypted bits each), and perform the entire computation bitwise. This way, however, no advantage is taken from the potential of arithmetic circuits.

In this paper, we address this shortcoming by means of efficient protocols for securely converting an integer into its binary representation. Such a protocol can be viewed as a new type of gate, next to the addition/subtraction gates and multiplication/division gates that are already known from [CDN01]. We call it the BITREP gate. So, on input $[\![x]\!]$, $0 \leq x < 2^m$, a BITREP gate outputs $[\![x_0]\!], \ldots, [\![x_{m-1}]\!]$, of course, without leaking any information on x.

A moment's thought will reveal that an efficient BITREP gate is only feasible if a cryptosystem such as Paillier is used as the underlying cryptosystem. For instance, the approach of [ST04], which is based on the homomorphic ElGamal cryptosystem, cannot lead to an efficient BITREP gate: given $[\![x]\!]$, $0 \leq x < 2^m$, extracting x cannot be done in time polynomial in m, as it involves solving a discrete log problem even when one knows the private key for $[\![\cdot]\!]$ (if $[\![x]\!] = (g^r, h^r g^x)$, one needs to recover x from g^x). For a Paillier ciphertext of the form $g^x r^N \bmod N^2$, there is no such contradiction as x can be recovered efficiently if one knows the private key.

As a related problem, we will also provide an efficient protocol for computing the least significant bits of an encrypted value, called an LSB gate. Clearly, a cascade of LSB gates can also be used to implement a BITREP gate, but our solutions for these gates will be of independent interest. The complexity of the LSB gate is independent of m, for input $[\![x]\!]$, $0 \leq x < 2^m$, under a mild restriction on m in terms of N, such as $m+100 < \log_2 N$ (in general, $m+\kappa+\log_2 n < \log_2 N$, for a security parameter κ).

As a further motivation for studying these problems we like to point out the following application. Consider a constrained device which must send a certain number of bits (ranging from a few hundred to a few thousand), which will be processed further as inputs to a secure computation. The obvious approach of sending a separate Paillier encryption for each of these many bits, however, may be completely infeasible. Given the BITREP gate, it suffices if the device packs the bits into one or more large integers, and sends these out using a single Paillier encryption per integer.

A particularly interesting case of this scenario can be seen in the context of biometric authentication. A tamper-resistant measuring device will obtain a biometric sample, e.g., an IrisCode® consisting of 512 bytes, which it sends out using a couple of Paillier encryptions. At the server side, the BITREP gate may then be applied to obtain the encrypted bits, which are subsequently used to evaluate securely whether the sample matches a stored (encrypted) biometric template, which was previously obtained during enrollment. In such a system, biometric details are never exposed in the clear, except in the measuring device during capture. The relevance of such an approach has been argued, e.g. in [DRS04,TG04,KAMR04].

Our Contributions

We present new n-party protocols for several variants of the problem of securely computing the binary representation of an integer value. In each case we make essential use of the Paillier cryptosystem (or variations thereof). The most general solution, which we call the BITREP gate, handles input values that can be *any* integer in $\{0, \ldots, N-1\}$, where N is an RSA modulus of $k = \log_2 N$ bits. The broadcast complexity of the BITREP gate ranges between $O(nk^2)$ and $O(n^2k^2 \log k)$ depending on which subprotocols are used (e.g., for the generation of jointly random bits).

Since k is large, ranging from 1024 to 2048 say, the cost of the BITREP gate is high even if the actual inputs are known to lie in a much smaller range than $\{0, \ldots, N-1\}$. Therefore, we also consider the case that the input values are from a limited range $\{0, \ldots, 2^m-1\}$, $m < k - \kappa - \log_2 n$, where κ is a security parameter to be set such that $2^{-\kappa}$ is negligible. In this case, we show how to reduce the broadcast complexity by a factor of k/m compared to the general case. The bound on m is not a severe restriction as typically $\kappa + \log_2 n$ is much smaller than k.

Finally, again for input values in $\{0, \ldots, 2^m-1\}$, $m < k - \kappa - \log_2 n$, we show how to extract the least significant bit in time *independent* of m (LSB gate). More generally, we show how to extract the ℓ least significant bits in time proportional to ℓ (LSBs gate). The broadcast complexity is reduced accordingly, by a factor k/ℓ compared to the general solution, independent of m.

Apart from these new protocols (which rely on a special application of $O(1)$ interval proofs), we also show how to integrate the security proofs for these new gates in the framework of [CDN01]. First, we observe that the security proof of [CDN01] actually achieves a *tight* reduction, namely that a distinguisher of the simulated vs. real protocol is transformed into a distinguisher for the underlying cryptosystem *without* loss in success probability. Secondly, the proof of [CDN01] is *modular* in the sense that a statistically indistinguishable simulation is given for each of the basic gates separately: e.g., the multiplication gate is simulated given inputs $[\![x]\!], [\![y]\!]$ and corresponding output $[\![xy]\!]$.

To retain the tightness and modularity, however, we need to state the security for a gate such as a BITREP gate in a particular way. It turns out to be impossible to simulate our BITREP gate given only input $[\![x]\!]$ and corresponding outputs $[\![x_0]\!], \ldots, [\![x_{m-1}]\!]$. The reason is that the protocol for the BITREP gate produces additional encryptions such as $[\![x_0 x_1]\!]$ and encryptions of other monomials as intermediate results, which cannot be computed by the simulator! This is an interesting phenomenon and we show how to extend the framework of [CDN01] to handle it. The LSB gate can easily be simulated, though, given just the input $[\![x]\!]$ and corresponding output $[\![x_0]\!]$.

We also highlight some applications of the new gates. A similar tradeoff as mentioned above for biometric authentication is possible in the context of electronic voting protocols, where one would like to minimize the effort required of the voter in casting an encrypted ballot (e.g., when the voter's client software needs to run on a simple mobile phone). This problem has already been considered in [DJ02], where incidentally Paillier encryption is used as well. Using a BITREP gate one gets an interesting alternative to [DJ02]. To vote for a single candidate x, $0 \leq x < 2^m$ say, one simply lets the voter release an encryption $[\![x]\!]$. To tally these votes we use the radix M representation of [CFSY96], where M is an integer larger than the number of voters. A vote x will thus contribute M^x to the final tally, represented as an integer in radix M. To compute this exponentiation securely, one first computes $[\![x_0]\!], \ldots, [\![x_{m-1}]\!]$ from $[\![x]\!]$, using the BITREP gate. Noting that $M^{x_0} = 1 + x_0(M-1)$ and so on, it follows that $[\![M^x]\!]$ can be computed securely, using $O(m)$ multiplication gates. The so-obtained encryptions $[\![M^x]\!]$ (for all voters) are then multiplied together and one gets, upon decryption, the election result in radix M representation.

Related Work

Independent of our work, the problem of securely computing the bits of an integer value has been studied recently by Damgård et al. [DFK+06]. An obvious difference is they consider the unconditional setting, whereas we consider the cryptographic model. This is reflected by the use of sharings (for an underlying linear secret sharing scheme) in their case vs. the use of encryptions (for an underlying threshold homomorphic cryptosystem) in our case.

The shared values as well as the shares and the arithmetic circuit in [DFK+06] are all defined over $\mathbb{F}_p$, for a prime p, where the security of the protocol does not depend on the size of p. Prime p can thus be chosen freely to fit an application. For example, to make things practical, one may use moderately large primes p of 64 bits say (e.g., to ensure that inputs of a reasonable size can be handled and, also, to easily exclude some failure events that happen with $O(1/p)$ probability). In our case, however, the arithmetic circuit will be defined over $\mathbb{Z}_N$, where N is the RSA modulus of a Paillier cryptosystem. Necessarily, the number N is therefore very large, say 1024 upto 2048 bits.

The protocol of [DFK$^+$06] for handling input values in $\{0, \ldots, p-1\}$ and our protocol for handling input values in $\{0, \ldots, N-1\}$ (the BITREP gate) follow the same pattern. The emphasis in [DFK$^+$06] is on constant round complexity (which is achieved by solving various subproblems in constant rounds). In this paper, we focus on techniques for handling inputs from a restricted range efficiently, to limit the consequences of the fact that all arithmetic is done modulo N, where N is necessarily large: informally, to extract the ℓ least significant bits, the complexity of our protocol is *independent* of the size of the gap $N - 2^\ell$.

The problem of securely computing the bits of an integer value has also been considered by Algesheimer et al. [ACS02], in the unconditional setting, but restricted to the passive case. Their approach follows a more complicated pattern, involving an addition circuit for securely adding n numbers bitwise (rather than just 2 numbers as we and [DFK$^+$06] do), which does not readily seem to extend to the problems and solutions considered in this paper and in [DFK$^+$06].

2 Preliminaries

In this section, we introduce some notation and present the basic tools that we need throughout the paper.

We assume the framework for secure computation based on threshold homomorphic cryptosystems of [CDN01], used with the Paillier cryptosystem. This framework allows one to securely evaluate arithmetic circuits, composed of several types of basic gates, as listed below. For simplicity, we assume that the parties $P_1, \ldots, P_n$ evaluating the circuit coincide with the parties running the underlying $(t+1, n)$-threshold Paillier cryptosystem. Here, t denotes the maximum number of statically, but actively corrupted parties tolerated by the circuit evaluation protocol. If a party fails to complete a step during any of the (sub)protocols (e.g., if a proof fails), then that party is simply discarded and the (sub)protocol is rerun by the remaining parties; we will not describe this explicitly for the protocols in this paper.

The bounds on t are as in previous papers. For $0 \leq t < n/2$, the case of a dishonest minority, robustness is achieved directly using the protocol of [CDN01]. For $n/2 \leq t < n$, the case of a dishonest majority, the protocol of [CDN01] can be extended in a modular way to achieve various degrees of fairness: the new property of "resource-fairness'", as introduced and studied in [GMPY06]), can be achieved under additional intractability assumptions such as the strong-RSA assumption; also, as described in [ST04], a strictly weaker form of fairness can be achieved without requiring additional assumptions, using a simple gradual release approach.

2.1 Paillier Cryptosystem

The Paillier cryptosystem is a probabilistic, additively homomorphic encryption scheme, known to be semantically secure under the *Decisional Composite Residuosity Assumption* [Pai99]. Several variations and generalizations of the basic Paillier cryptosystem have been proposed since, see, e.g., [FPS00,DJ01,CS02,DJ03].

Below, we summarize the threshold variant of a generalized Paillier cryptosystem, as introduced in [DJ01, Section 4.1], but any other variant providing threshold decryption can be used as well for our purposes.

The public key consists of an RSA modulus $N = pq$ of bit length k (security parameter), where p, q are safe primes, $p = 2p' + 1$ and $q = 2q' + 1$. The set of plaintexts is given by the additive group $\mathbb{Z}_{N^s}$, and an encryption of a message $x \in \mathbb{Z}_{N^s}$ takes the form $[\![x]\!] = (N+1)^x r^{N^s} \bmod N^{s+1}$ for random $r \in \mathbb{Z}^*_{N^{s+1}}$ (case $s = 1$ corresponds to the basic Paillier cryptosystem).

The private key is given by the unique value $d \in \mathbb{Z}_{\tau N^s}$ satisfying $d = 0 \bmod \tau$ and $d = 1 \bmod N^s$, where $\tau = p'q'$. In the threshold case, polynomial shares d_i for $i = 1, \ldots, n$ of d are generated and distributed parties $P_1, \ldots, P_n$. Decryption of a ciphertext $[\![x]\!]$ is done by means of a (t, n)-threshold decryption protocol requiring the cooperation of t or more parties, $1 \leq t \leq n$. We refer to [DJ01] for further details, noting that it is also proved that, given a ciphertext and the corresponding plaintext, the decryption protocol can be simulated statistically indistinguishable for an active, static adversary corrupting at most t parties.

Throughout this paper, the results are described for the case $s = 1$, as in the original Paillier cryptosystem, such that the plaintext space is simply the additive group $\mathbb{Z}_N$ (but the results can readily be extended to the case $s > 1$). In this case, the cryptosystem is additively homomorphic over $\mathbb{Z}_N$: given $[\![x]\!]$ and $[\![y]\!]$ with $x, y \in \mathbb{Z}_N$ we have $[\![x + y]\!] = [\![x]\!][\![y]\!]$, where multiplication of ciphertexts is done modulo N^2. Note that this implies that, for any $c \in \mathbb{Z}_N$, $[\![cx]\!] = [\![x]\!]^c \bmod N^2$.

2.2 Efficient Proofs

In order to withstand active attacks, we use several standard types of zero-knowledge proofs (Σ-protocols). We will assume the *random oracle model*, so that all the proofs are non-interactive and can be simulated easily. We use standard proofs for proving plaintext knowledge, equality of plaintexts, and that a plaintext is a bit for given Paillier encryptions.

In particular, we will use efficient *range proofs* by which a party that generated a Paillier encryption $[\![x]\!]$ can prove that x belongs to given interval using $O(1)$ modular exponentiations only. Although the hidden constant is quite large, the $O(1)$ methods certainly pay off for intervals of length 2^{100} and up, as we need (hence, the size of the non-interactive versions of these proofs is $O(k)$ for security parameter k, in the random oracle model). Efficient proofs for showing that a committed integer value belongs to a given interval were introduced in [Bou00], and later refined in [Lip03]. The underlying integer commitment scheme of [FO97,DF02] relies on the *strong RSA assumption*. By proving equality of such a committed value and an encrypted value (see, e.g., [DJ02]), one can thus prove that an encrypted value belongs to a given interval.

2.3 Basic Gates and Circuits

In this section, we briefly describe the secure subprotocols (gates and circuits) we need for our protocols. The distinction between gates and circuits may be somewhat arbitrary.

We describe the standard gates for multiplication and inversion modulo N, followed by two gates (or circuits) for jointly generating random values. Finally, we describe circuits for the bitwise operations used in our protocols. These are the basic arithmetic operators for addition and subtraction of integers, and the basic relational operators for comparison of integers.

The exact details of these gates and circuits are immaterial to the validity of our constructions, but we do give an indication of the broadcast and round complexities.

Multiplication and Inversion Gates The general multiplication gate developed in [CDN01] allows n parties to compute an encryption $[\![xy]\!]$ given encryptions $[\![x]\!]$ and $[\![y]\!]$, where $x, y \in \mathbb{Z}_N$, in a constant number of rounds.

In case one of the inputs, say input x, is private to one of the parties, say party P_i, the following simplified multiplication protocol can be used, requiring no interaction at all. Party P_i computes $[\![xy]\!]$ directly from $[\![y]\!]$ using the homomorphic property $[\![xy]\!] = [\![y]\!]^x$. Party P_i also generates a Σ-proof showing that it computed $[\![xy]\!]$ correctly w.r.t. $[\![x]\!]$ and $[\![y]\!]$. We will refer to this gate as a *private-multiplier gate*.

An inversion gate allows n parties to compute an encryption $[\![x^{-1}]\!]$ given encryption $[\![x]\!]$, with $x \in \mathbb{Z}_N^*$, in a constant number of rounds [CDN01].

The broadcast complexity is $O(nk)$ bits for a multiplication gate and an inversion gate, and $O(k)$ bits for a private-multiplier gate.

Random Invertible Element Gates All parties choose a random element $r_i \in_R \mathbb{Z}_N^*$ and broadcast an encryption $[\![r_i]\!]$ together with a Σ-proof for knowledge of the plaintext r_i. Finally, $[\![r]\!] = [\![\sum_{i=1}^n r_i]\!]$ is publicly computed. Note that this gate fails with negligible probability.

The broadcast complexity is $O(nk)$ bits.

Random-Bit Gates We list three protocols for jointly generating encrypted random bits. Each protocol starts the same. For $i = 1, \ldots, n$, party P_i generates a uniformly random bit $b_i \in \{0, 1\}$ and broadcasts $[\![b_i]\!]$ together with a Σ-proof to show knowledge of b_i and that b_i is indeed a bit. To combine bits $[\![b_i]\!]$ into a joint random bit $[\![b]\!]$, with $b = \oplus_{i=1}^n b_i$, we mention three options:

- use of an unbounded fan-in multiplication gate to compute $[\![b]\!]$ in constant rounds, see [CDN01];
- use of $O(n)$ multiplication gates to compute $[\![b]\!]$ in $O(\log n)$ rounds;
- use of $O(n)$ private-multiplier gates to compute $[\![b]\!]$ in $O(n)$ rounds.

The broadcast complexity is $O(n^2k)$ bits for the first two options (but the hidden constant is significantly higher for the constant rounds protocol), and $O(nk)$ bits for the last option. Note that the joint random bits can be computed in preprocessing, if so desired, creating an opportunity to use more rounds for a lower broadcast complexity (and for a lower computational complexity too).

Addition and Subtraction Circuits Given encrypted bit representations $[\![x_0]\!], \ldots, [\![x_{m-1}]\!]$ and $[\![y_0]\!], \ldots, [\![y_{m-1}]\!]$ of two numbers x, y, an addition circuit essentially computes the bits of $x+y$, given by $[\![z_0]\!], \ldots, [\![z_{m-1}]\!], [\![c_{m-1}]\!]$ as follows:

$$\begin{aligned}
&z_i = x_i + y_i + c_{i-1} - 2c_i \\
&c_{-1} = 0, \quad c_i = x_i y_i + x_i c_{i-1} + y_i c_{i-1} - 2x_i y_i c_{i-1}.
\end{aligned}$$

A similar circuit can be used for subtraction.

Depending on the scenario such a circuit can be refined in various ways. We mention two extreme options. One can optimize for broadcast complexity and use $O(m)$ rounds to compute an addition with broadcast complexity $O(mnk)$ bits. At the other extreme, one can optimize for round complexity and use $O(1)$ rounds as shown in [DFK+06]; their $O(1)$-depth circuit for bitwise addition can also be used in the threshold homomorphic setting, resulting in a broadcast complexity of $O(m \log mnk)$ bits.

Equality and Comparison Circuits Let $[C]$ denote the Iverson bracket defined by $[C] = 1$ if $C \Leftrightarrow$ true and $[C] = 0$ otherwise. Given encrypted bit representations $[\![x_0]\!], \ldots, [\![x_{m-1}]\!]$ and $[\![y_0]\!], \ldots, [\![y_{m-1}]\!]$ of two numbers x, y, equality and comparison circuits compute $[\![[x = y]]\!]$ or $[\![[x < y]]\!]$, respectively. A $O(\log m)$-depth circuit for equality is straightforward, leading to round complexity of $O(\log m)$ and broadcast complexity of $O(mnk)$ bits. Interestingly, $O(1)$ rounds and $O(mnk)$ bits are also achievable as shown in [DFK+06]; again their methods carry over to the threshold homomorphic setting.

3 LSB Gate

We present a protocol for securely computing the least significant bit, which on input $[\![x]\!]$, outputs an encryption $[\![x_0]\!]$. To obtain a particularly efficient solution, we assume that x is a bounded value, that is, $0 \leq x < 2^m$ where the value of m is restricted as a function of N, the number of parties n, and a security parameter κ. The parameter κ is chosen such that $2^{-\kappa}$ is negligible. The restriction on m is that $m + \kappa + \log_2 n < \log_2 N$. In practice this is not a severe restriction. For example, if N is a 1024-bit modulus, $\kappa = 100$, and $n = 16$, then m is bounded above by 920.

3.1 Protocol

The idea is to jointly generate a random value $[\![r]\!]$ and to decrypt $[\![x + r]\!]$ such that (i) $y = x + r$ is statistically indistinguishable from random and (ii) $[\![x_0]\!]$ can be recovered from y_0 and $[\![r_0]\!]$. To this end, the random value r will be generated in the form $r = r_0 + 2r_*$, where r_0 is a bit and r_* is an integer value from a sufficiently large range.

LSB Gate

1. The parties jointly generate a random bit $[\![r_0]\!]$, using a random-bit gate. In parallel, each party P_i chooses $r_{*,i} \in_R \{0, \ldots, 2^{m+\kappa-1} - 1\}$ and broadcasts $[\![r_{*,i}]\!]$ accompanied with a range proof that the encryption is correctly formed. The encryption $[\![r_*]\!]$ with $r_* = \sum_{i=1}^{n} r_{*,i}$ is publicly computed.
2. The encryption $[\![x]\!][\![r_0]\!][\![r_*]\!]^2$ is formed and jointly decrypted to reveal the value $y = x + r$, where $r = r_0 + 2r_*$.
3. The output is $[\![r_0 \oplus y_0]\!]$, which can be computed publicly from $[\![r_0]\!]$ and y_0, as $r_0 \oplus y_0 = r_0 + y_0 - 2r_0 y_0$.

The broadcast complexity incurred by the range proofs in the first step of the protocol is limited to $O(nk)$ bits, assuming an efficient range proof is used, as mentioned in Section 2.2. The broadcast complexity of the entire protocol depends on the broadcast complexity of the random-bit gate used for generating $[\![r_0]\!]$, and varies between $O(n^2k)$ bits (and $O(1)$ round complexity) and $O(nk)$ bits (and $O(n)$ round complexity).

We note that once x_0 is computed, the next bit of x can be computed by applying the protocol to $[\![x_*]\!]$, with $x_* = (x - x_0)/2$. Indeed, the homomorphic property implies $[\![x_*]\!] = ([\![x]\!]/[\![x_0]\!])^{1/2}$, where $1/2 = (N+1)/2$ is the inverse of 2 modulo N. This way all of the bits of x can be recovered.

3.2 Security

The value output by the protocol is correct because $y_0 = x_0 \oplus r_0$, as the addition $y = x + r$ is computed over the integers (the limited size of x and r ensures that $0 \le x + r < N$).

Next, we show that the protocol can be simulated, assuming that the random-bit gate can be simulated when given an output $[\![\tilde{r}_0]\!]$. Of course, the bit $\tilde{r}_0$ should be distributed uniformly.

Theorem 1. *On input $[\![x]\!]$ and $[\![x_0]\!]$, where $0 \le x < 2^m$, the* LSB *gate can be simulated in a statistically indistinguishable manner.*

Proof. Let $x_* = (x - x_0)/2$. Then $[\![x_*]\!] = ([\![x]\!]/[\![x_0]\!])^{1/2}$.

To argue that no additional information on x is leaked we present the following simulation of the protocol. The simulation takes as input encryptions $[\![x]\!]$ and $[\![x_0]\!]$. Given this information, the simulator is able to generate a complete transcript for the protocol, for which the distribution is exactly the same as in real executions of the protocol. Note that we have to ensure that the simulator "knows" the plaintext for the joint decryption step in the middle of the protocol, as the simulator for the threshold decryption protocol needs both a ciphertext and the corresponding plaintext as input.

Assume w.l.o.g. that parties $P_1, \ldots, P_t$ are corrupted. The simulator chooses $\tilde{y}_0 \in_R \{0,1\}$. We use the simulator for the random-bit gate to obtain a simulation for output $[\![\tilde{r}_0]\!]$ with $\tilde{r}_0 = x_0 \oplus \tilde{y}_0$. This encryption can be computed from the given encryption $[\![x_0]\!]$, using $\tilde{y}_0$. In parallel, the simulator lets the adversary run the first step of the protocol for parties $P_1, \ldots, P_t$, each time rewinding the proofs of knowledge used to extract the values $r_{*,1}, \ldots, r_{*,t}$. Subsequently, the simulator runs the first step of the protocol for parties $P_{t+1}, \ldots, P_{n-1}$ as in the real protocol, resulting in the values $r_{*,t+1}, \ldots, r_{*,n-1}$. For party P_n, however, the simulator first generates $s \in_R \{0, \ldots, 2^{m+\kappa-1}-1\}$, and sets $\tilde{r}_{*,n} = s - x_* - (x_0 + \tilde{r}_0 - \tilde{y}_0)/2$. Then $[\![\tilde{r}_{*,n}]\!]$ can be computed using $[\![x_0]\!]$ and $[\![x_*]\!]$. The range proof for $[\![\tilde{r}_{*,n}]\!]$ is generated using the simulator for these proofs. Finally, the simulator sets $\tilde{r}_* = \sum_{i=1}^{n-1} r_{*,i} + \tilde{r}_{*,n}$.

As a result, we have that

$$[\![x+\tilde{r}_0+2\tilde{r}_*]\!] = [\![x+\tilde{r}_0+2\sum_{i=1}^{n-1} r_{*,i}+2s-2x_*-(x_0+\tilde{r}_0-\tilde{y}_0)]\!] = [\![\tilde{y}_0+2(\sum_{i=1}^{n-1} r_{*,i}+s)]\!],$$

for which the simulator knows the decryption, as required.

By construction, the values are all consistent. We need to argue that all the probability distributions for these values are correct too. Clearly, the values for parties $P_1, \ldots, P_{n-1}$ follow the right distribution. Since $\tilde{y}_0$ is a uniformly random bit, so is $\tilde{r}_0$. Finally, the distribution of $r_{*,n} = s - x_* - (x_0 + \tilde{r}_0 - \tilde{y}_0)/2$ is statistically indistinguishable from the uniform distribution on $\{0, \ldots, 2^{m+\kappa-1}-1\}$, because $s \in_R \{0, \ldots, 2^{m+\kappa-1}-1\}$ and the term $x_* + (x_0 + \tilde{r}_0 - \tilde{y}_0)/2 = x_* + x_0(1-\tilde{y}_0)$ is much smaller than it (bounded above by 2^m). More precisely, the statistical distance is bounded by $2^{-\kappa+1}$ (see Appendix A).

4 LSBs Gate

Next, we consider the more general case of extracting the ℓ least significant bits of x rather than just the least significant one. We describe the protocol for the case that *all* of the bits of x are computed (case $\ell = m$); the case $\ell < m$ can be handled by combining the techniques of this section and the previous section.

4.1 Protocol

Let $m + \kappa + \log_2 n < \log_2 N$, as before. On input $[\![x]\!]$, where $0 \le x < 2^m$, the following protocol computes $[\![x_0]\!], \ldots, [\![x_{m-1}]\!]$ securely. The idea is to jointly generate a random value $[\![r]\!]$ and to decrypt $[\![x+r]\!]$ such that (i) $y = x + r$ is statistically indistinguishable from random and (ii) $[\![x_0]\!], \ldots, [\![x_{m-1}]\!]$ can be recovered from $y_0, \ldots, y_{m-1}$ and $[\![r_0]\!], \ldots, [\![r_{m-1}]\!]$. To this end, the value r will be generated in the form $r = \sum_{j=0}^{m-1} r_j 2^j + 2r_*$, where $r_0, \ldots, r_{m-1}$ are bits and r_* is an integer value from a sufficiently large range.

For technical reasons, we will actually compute $y = x - r$ in (i) and use an addition circuit to perform step (ii).

<u>LSBs Gate</u>

1. The parties jointly generate random bits $[\![r_0]\!], \ldots, [\![r_{m-1}]\!]$, using m random-bit gates. In parallel, each party P_i chooses $r_{*,i} \in_R \{0, \ldots, 2^{m+\kappa-1} - 1\}$ and broadcasts $[\![r_{*,i}]\!]$ accompanied with a range proof that the encryption is correctly formed. The encryption $[\![r_*]\!]$ with $r_* = \sum_{i=1}^{n} r_{*,i}$ is publicly computed.
2. The encryption $[\![x - r]\!]$ is formed and jointly decrypted to reveal the signed value $y = x - r \in (-n/2, n/2)$, where $r = \sum_{j=0}^{m-1} r_j 2^j + r_* 2^m$. The signed value y is computed such that $y \equiv x - r \pmod{n}$.
3. Let $y_0, \ldots, y_{m-1}$ denote the binary representation of $y \bmod 2^m$. An addition circuit for inputs $y_0, \ldots, y_{m-1}$ (public) and $[\![r_0]\!], \ldots, [\![r_{m-1}]\!]$ is used to produce an output of m encrypted bits (ignoring the final carry bit, hence computing modulo 2^m).

Note that the probability that $y \geq 0$ at step 2 is negligible.

The broadcast complexity of the protocol depends on the broadcast complexity of the random-bit gate used for generating $[\![r_0]\!], \ldots, [\![r_{m-1}]\!]$, and varies between $O(mn^2k)$ bits (and $O(1)$ round complexity) and $O(mnk)$ bits (and $O(n)$ round complexity).

4.2 Security

The main proof obligation is to show that the protocol can be simulated. We would like to do so given just a matching pair of inputs and outputs, which in this case consists of the encryptions $[\![x]\!]$ and $[\![x_0]\!], \ldots, [\![x_{m-1}]\!]$. Unfortunately, such a simulation will not succeed for the LSBs protocol because the protocol is releasing *additional* encryptions apart from the encrypted bits of x. The problem lies in the addition circuit, as used in the final step of the protocol.

The problem is that an addition circuit (see Section 2.3) releases encrypted carry bits, next to the encrypted output bits. Even with full knowledge of r, these encrypted carry bits can, in general, not be computed from $[\![x_0]\!], \ldots, [\![x_{m-1}]\!]$, as this would imply that encryptions such as $[\![x_0 x_1]\!]$, or other encrypted monomials, can be computed in polynomial time. What is more, the number of monomials that are possibly needed in a simulation is equal to 2^m, hence exponential m.

We observe, however, that to prove simulatability in the framework of [CDN01] it suffices to perform a simulation for input/output pairs of a special form. This is a consequence of the fact that the security proof of [CDN01] (full version) centers around the construction of the so-called YAD^b distribution, which is defined as a function of an encrypted bit $[\![b]\!]$.

In terms of the YAD^b distribution, the structure of the security proof is as follows, following an ideal/real approach. The YAD^0 distribution is identical to the distribution in the ideal case, whereas the YAD^1 distribution is statistically indistinguishable from the distribution in the real case. Consequently, if an adversary is able to distinguish the ideal/real cases, it follows that the adversary is able to distinguish the YAD^0 distribution from the YAD^1 distribution. But the choice between these two distributions is entirely determined by the value of the encrypted bit b. Hence, a distinguisher for the ideal/real cases implies a distinguisher for the underlying cryptosystem, and it does so in a *tight* way (without loss in success probability for the distinguisher).

The special form for the input/output pairs is given by $[\![\tilde{x}]\!] = [\![(1-b)x+bx']\!]$, where x and x' are given in the clear, but bit b is only given as an encryption $[\![b]\!]$. The values x and x' correspond to the values arising in the YAD^0 case and the YAD^1 case, respectively, and are *both* known to the simulator. The x-values correspond to fake values used to set up a consistent simulated view ($b = 0$) and the x'-values correspond to the values used to set up a consistent real view ($b = 1$).

Thus, the security is stated in a less general way, but still sufficiently general to match the (adapted) framework of [CDN01]. Stating the security of a gate (or, sub-circuit) in this way allows one to capture the security in a modular way, while retaining the tightness of the overall reduction. Clearly, this idea is more widely applicable, beyond the gates we are considering in this paper.

Theorem 2. *Given input values x, x' with $0 \leq x, x' < 2^m$ and an encryption $[\![b]\!]$ with $b \in \{0, 1\}$, the* LSBs *gate can be simulated statistically indistinguishably for input $[\![\tilde{x}]\!] = [\![(1-b)x + bx']\!]$.*

Proof. The goal is to generate a simulated transcript for input $[\![\tilde{x}]\!] = [\![(1 - b)x + bx']\!]$, which is to the adversary statistically indistinguishable from a real transcript. The values of x, x' and $[\![b]\!]$ are available to the simulator.

Assume w.l.o.g. that parties $P_1, \ldots, P_t$ are corrupted. Pick $\tilde{r}_0, \ldots, \tilde{r}_{m-1} \in_R \{0, 1\}$ and simulate the generation of these random bits. The simulator extracts/generates the values $r_{*,i}$ for the corrupted/honest parties P_i ($1 \leq i < n$). The simulator chooses $s \in_R \{0, \ldots, 2^{m+\kappa} - 1\}$ and sets $r_{*,n} = s - x - \sum_{j=0}^{m-1} \tilde{r}_j 2^j$ and $r'_{*,n} = s - x' - \sum_{j=0}^{m-1} \tilde{r}_j 2^j$. The distributions will be statistically close to the uniform distribution on $\{0, \ldots, 2^{m+\kappa} - 1\}$.

By construction, the simulator knows how to decrypt $x - r$ and $x' - r'$, where $r = \sum_{j=0}^{m-1} \tilde{r}_j 2^j + r_* 2^m$ and $r' = \sum_{j=0}^{m-1} \tilde{r}_j 2^j + r'_* 2^m$, and where $r_* = \sum_{i=1}^{n-1} r_{*,i} + r_{*,n}$ and $r'_* = \sum_{i=1}^{n-1} r_{*,i} + r'_{*,n}$

To complete the proof we need to assign the correct values to all the wires in the addition circuit, consistent with either $b = 0$ or $b = 1$. In both cases, the first input to the addition circuit is $s_0, \ldots, s_{m-1}$. The second input is either $r_0, \ldots, r_{m-1}$ or $r'_0, \ldots, r'_{m-1}$, and the corresponding output is $x_0, \ldots, x_{m-1}$ or $x'_0, \ldots, x'_{m-1}$. Now, for each wire, two values can be computed from these values. If values v and v' are thus computed for such a wire, we assign the encryption $[\![(1 - b)v + bv']\!]$ to the wire.

This leads to a consistent assignment of encryptions to all of the wires. And, therefore the simulators for the multiplication gates constituting the addition circuit can be used to complete the simulation.

The values generated by the simulator are all consistent. Moreover, the distribution of $-s$ is statistically indistinguishable from the value of y used in the real protocol.

5 BITREP Gate

In this section we consider the case that x is any value in the range $[0, N)$, where N is the value of the Paillier modulus. We first present a protocol for jointly generating a random value $r \in_R [0, N)$, given by its bits $[\![r_0]\!], \ldots, [\![r_{m-1}]\!]$, with m denoting the bit length of N. Subsequently, we present the BITREP gate, which converts $[\![x]\!]$ into $[\![x_0]\!], \ldots, [\![x_{m-1}]\!]$.

The protocol for generating a random value $r \in_R [0, N)$ uses the basic protocol for jointly generating m random bits between parties $P_1, \ldots, P_n$. We then test whether the integer represented by these m bits is in the range $[0, N)$:

1. The parties jointly generate random bits $[\![r_0]\!], \ldots, [\![r_{m-1}]\!]$, using m random-bit gates.
2. A comparison circuit for encrypted inputs $[\![r_0]\!], \ldots, [\![r_{m-1}]\!]$ and public inputs $N_0, \ldots, N_{m-1}$, denoting the bits of N, is used to compute $[\![[r < N]]\!]$, where $r = \sum_{j=0}^{m-1} r_j 2^j$.
3. The value $[\![[r < N]]\!]$ is decrypted to see if r is in range; if not, go back to the first step.

The average number of iterations is bounded above by 2.

The protocol for converting $[\![x]\!]$ into $[\![x_0]\!], \ldots, [\![x_{m-1}]\!]$, where $0 \le x < N$, now runs as follows.

<u>BITREP Gate</u>

1. The parties generate encrypted bits $[\![r_0]\!], \ldots, [\![r_{m-1}]\!]$ of a random number $0 \le r < N$.
2. The parties compute $[\![x]\!] \prod_{j=0}^{m-1} [\![r_j]\!]^{2^j}$ and perform a threshold decryption to obtain $y = x + r \bmod N$, $0 \le y < N$.
3. Using a subtraction circuit with $y_0, \ldots, y_{m-1}$ and $[\![r_0]\!], \ldots, [\![r_{m-1}]\!]$ as inputs, the parties determine the bit representation $[\![z_0]\!], \ldots, [\![z_m]\!]$ of the value $z = x$ or $z = x - N$, where z_m is a sign bit.
4. The parties reduce the value of z modulo N, by adding Nz_m to z using an addition circuit with inputs $[\![(Nz_m)_0]\!], \ldots, [\![(Nz_m)_{m-1}]\!]$ and $[\![z_0]\!], \ldots, [\![z_{m-1}]\!]$.

Note that the equality $y = x + r$ holds in $\mathbb{Z}_n$ but not necessarily in $\mathbb{Z}$. But if $y \ne x + r$ over the integers, then it follows that $y = x + r - N$ must hold over the integers, since $0 \le x < N$ and $0 \le r < N$. In step 3, the case $z = x$ occurs exactly when $y = x + r$ over the integers, and the case $z = x - N$ occurs when $y = x + r - N$.

The security is proved similar as in the previous section.

Theorem 3. *Given input values x, x' with $0 \le x, x' < N$ and an encryption $[\![b]\!]$ with $b \in \{0, 1\}$, the* BITREP *gate can be simulated statistically indistinguishably for input $[\![\tilde{x}]\!] = [\![(1 - b)x + bx']\!]$.*

6 Applications

The result of [CDN01] shows how to efficiently evaluate arithmetic circuits composed of addition/subtraction gates and multiplication/division gates defined over $\mathbb{Z}_N$. This way large numbers can be handled, practically independent of the size of the numbers. This contrasts favorably with approaches based on Boolean circuits, where arithmetic is done in a bitwise fashion. However, as argued in the introduction, the potential of arithmetic circuits is limited when some (inherently) bitwise operations are required as well. Without binary conversion gates, one is forced to perform the entire computation using a (large) Boolean circuit.

The relational operators such as $<$ and $=$ are typically handled by representing the numbers in binary form. Another important example is exponentiation, where one wishes to compute $[\![x^y]\!]$ securely, given $[\![x]\!]$ and $[\![y]\!]$. Using a method of repeated squaring one may compute $[\![x^y]\!]$ using $O(m)$ multiplication gates, once $[\![y]\!]$ is converted into binary, given by its encrypted bits $[\![y_0]\!], \ldots, [\![y_{m-1}]\!]$, say.

As an interesting application we close this paper with a discussion of private biometric authentication. This topic received quite some attention during the last years, see e.g., [DRS04,TG04,KAMR04]. The goal of private biometric authentication is to identify or authenticate people based on their physical characteristics (fingerprint, iris, ...) without revealing any information on these personal characteristics to the verifier or an attacker. This problem has been investigated in the information theoretical setting by several authors [TG04,DRS04]. They gave general constructions (using "helper data" and "fuzzy extractors" resp.). At the same time, it was shown that perfect privacy cannot be achieved from an information theoretic point of view. It is therefore natural to explore whether a full privacy preserving and efficient biometric authentication scheme can be constructed in the cryptographic setting. Below, we briefly describe how this is achieved.

The heart of the system is formed by a set of servers which correspond to the parties $P_1, \ldots, P_n$ sharing the private key for a threshold Paillier cryptosystem. These servers will match encrypted biometric templates as obtained during enrollment of the users against encrypted biometric templates as measured by sensors as part of an authentication protocol. Since the sensors are typically lightweight devices, the goal is to minimize the computational load for the sensors.

Authentication of a user will succeed if the biometric template $Y = (y_1, \ldots, y_m)$ measured by a sensor is sufficiently close to the biometric template $X = (x_1, \ldots, x_m)$ obtained during enrollment. For instance, assuming that the biometric templates are actually bit strings in $\{0, 1\}^m$, a possible similarity measure is the Hamming distance between the bit strings: if $d_H(X, Y) < T$, where T is a predetermined threshold, then X and Y are said to match.

The BITREP gate can now be used as follows. To minimize the work for the sensor we let the sensor first convert the measured biometric template Y to an integer $y \in \{0, \ldots, 2^m - 1\}$, using an obvious mapping. The sensor is then required to release the encrypted value $[\![y]\!]$ together with the claimed identifier for the person being authenticated. At the server side, the encrypted bits of

Y are recovered using a BITREP gate (or, an LSBs gate if appropriate), before running a secure matching protocol on the encrypted bits of X and Y.

Acknowledgements We would like to thank the anonymous referees for their helpful comments.

References

[ACS02] J. Algesheimer, J. Camenisch, and V. Shoup. Efficient computation modulo a shared secret with application to the generation of shared safe-prime products. In *Advances in Cryptology—CRYPTO '02*, volume 2442 of *Lecture Notes in Computer Science*, pages 417–432, Berlin, 2002. Springer-Verlag.

[Bou00] F. Boudot. Efficient proofs that a committed number lies in an interval. In *Advances in Cryptology—EUROCRYPT '00*, volume 1807 of *Lecture Notes in Computer Science*, pages 431–444, Berlin, 2000. Springer-Verlag.

[CDN01] R. Cramer, I. Damgård, and J.B. Nielsen. Multiparty computation from threshold homomorphic encryption. In *Advances in Cryptology—EUROCRYPT '01*, volume 2045 of *Lecture Notes in Computer Science*, pages 280–300, Berlin, 2001. Springer-Verlag. Full version `eprint.iacr.org/2000/055`, October 27, 2000.

[CFSY96] R. Cramer, M. Franklin, B. Schoenmakers, and M. Yung. Multi-authority secret ballot elections with linear work. In *Advances in Cryptology—EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 72–83, Berlin, 1996. Springer-Verlag.

[CS02] R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *Advances in Cryptology—EUROCRYPT '02*, volume 2332 of *Lecture Notes in Computer Science*, pages 45–64, Berlin, 2002. Springer-Verlag.

[DF02] I. Damgård and E. Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In *Advances in Cryptology—ASIACRYPT '02*, volume 2501 of *Lecture Notes in Computer Science*, pages 125–142, Berlin, 2002. Springer-Verlag.

[DFK+06] I. Damgård, M. Fitzi, E. Kiltz, J.B. Nielsen, and T. Toft. Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In *Proc. 3rd Theory of Cryptography Conference, TCC 2006*, volume 3876 of *Lecture Notes in Computer Science*, pages 285–304, Berlin, 2006. Springer-Verlag.

[DJ01] I. Damgård and M. Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *Public Key Cryptography—PKC '01*, volume 1992 of *Lecture Notes in Computer Science*, pages 119–136, Berlin, 2001. Springer-Verlag.

[DJ02] I. Damgård and M. Jurik. Client/server tradeoffs for online elections. In *Public Key Cryptography—PKC '02*, volume 2274 of *Public Key Cryptography—PKC '*, pages 125–140, Berlin, 2002. Springer-Verlag.

[DJ03] I. Damgård and M. Jurik. A length-flexible threshold cryptosystem with applications. In *ACISP 2003*, volume 2727 of *Lecture Notes in Computer Science*, pages 350–364, Berlin, 2003. Springer-Verlag.

[DRS04] Y. Dodis, M. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *Advances in Cryptology—EUROCRYPT '04*, volume 3027 of *Lecture Notes in Computer Science*, pages 523–540, Berlin, 2004. Springer-Verlag.

[FO97] E. Fujisaki and T. Okamoto. Statistical zeroknowledge protocols to prove modular polynomial relations. In *Advances in Cryptology—CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 16–30, Berlin, 1997. Springer-Verlag.

[FPS00] P.-A. Fouque, G. Poupard, and J. Stern. Sharing decryption in the context of voting or lotteries. In *Financial Cryptography 2000*, volume 1962 of *Lecture Notes in Computer Science*, pages 90–104, Berlin, 2000. Springer-Verlag.

[GMPY06] J. Garay, P. MacKenzie, M. Prabhakaran, and K. Yang. Resource fairness and composability of cryptographic protocols. In *Proc. 3rd Theory of Cryptography Conference, TCC 2006*, volume 3876 of *Lecture Notes in Computer Science*, pages 404–428, Berlin, 2006. Springer-Verlag.

[KAMR04] F. Kerschbaum, M.J. Atallah, D. M'Raïhi, and J.R. Rice. Private fingerprint verification without local storage. In *Proceedings of the first International Conference on Biometric Authentication*, volume 3072 of *Lecture Notes in Computer Science*, pages 387–394, Berlin, 2004. Springer-Verlag.

[Lip03] H. Lipmaa. On diophantine complexity and statistical zero-knowledge arguments. In *Advances in Cryptology—ASIACRYPT '03*, volume 2894 of *Lecture Notes in Computer Science*, pages 398–415, Berlin, 2003. Springer-Verlag.

[Pai99] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology—EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238, Berlin, 1999. Springer-Verlag.

[ST04] B. Schoenmakers and P. Tuyls. Practical two-party computation based on the conditional gate. In *Advances in Cryptology—ASIACRYPT '04*, volume 3329 of *Lecture Notes in Computer Science*, pages 119–136, Berlin, 2004. Springer-Verlag.

[TG04] P. Tuyls and J. Goseling. Capacity and examples of template protecting biometric authentication systems. In *Proceedings of Biometric Authentication Workshop*, volume 3087 of *Lecture Notes in Computer Science*, pages 158–170, Berlin, 2004. Springer-Verlag.

A Statistical Distance

We use elementary results on statistical distance in our proofs. As an illustration we prove the following one.

Lemma 1. *Let M and K be positive integers, $M \leq K$. Let random variable X take on values in $\{0, \ldots, M-1\}$, and let random variable U be uniform on $\{0, \ldots, K-1\}$. Then $\Delta(U, X+U) \leq (M-1)/K$, and this upper bound is tight.*

Proof. For any w we have that $\Pr[X+U=w] = \sum_{a=0}^{M-1} \Pr[X=a]\Pr[U=w-a]$, hence that $0 \leq \Pr[X+U=w] \leq 1/K$, and, if $M-1 \leq w < K$, that

$\Pr[X + U = w] = 1/K$. Thus,

$$\begin{aligned}
&\Delta(U, X+U) = \tfrac{1}{2}\textstyle\sum_{w=0}^{M+K-2} |\Pr[U = w] - \Pr[X + U = w]| \\
&\leq \tfrac{1}{2}\left(\textstyle\sum_{w=0}^{M-2} |1/K - 0| + \sum_{w=M-1}^{K-1} |1/K - 1/K| + \sum_{w=K}^{M+K-2} |0 - 1/K|\right) \\
&= (M-1)/K.
\end{aligned}$$

Take $X = M - 1$ (constant) to conclude that this bound is tight.

Hence, $\Delta(U, X + U)$ is small if $K \gg M$. For instance, if K is exponential in a security parameter k and M is polynomial in k, then the statistical distance is negligible in k. In particular, if $M = 2^m$ and $K = 2^{m+\kappa}$, then $\Delta < 2^{-\kappa}$.