Secure and Efficient Constructions of Hash, MAC and PRF for Mobile Devices

Yan Zhu, Shanbiao Wang
Peking University,
Beijing, 100871, China
{yan.zhu,sbwang}@pku.edu.cn

Di Ma
University of Michigan-Dearborn,
Dearborn, Michigan, 48128
dmadma@umich.edu

Hongxin Hu
Delaware State University,
Dover, Delaware, 19901
hxhu@asu.edu

Gail-Joon Ahn
Arizona State University,
Tempe, Arizona, 85287
gahn@asu.edu

***Abstract*—Numerous cryptographic techniques have been developed to be used on mobile devices for various security and privacy protections. However, these cryptographic primitives, working under different mathematical assumptions, tend to become more and more complex and intricate, which makes it increasingly more difficult for proper implementation and management. Thus, it is desired to simplify management and improve efficiency by means of designing a general function family to meet a variety of security requirements. In this paper, we present such a family of square functions, including SqHash, SqMAC and SqPRF, based on a specially truncated function (MSB or LSB). We further improve the efficiency of these algorithms by using "circular convolution with carry bits" which makes parallel processing possible. We prove the security of these functions based on the privacy in hidden number problem and hard-core predicate of one-way function. We also show that the proposed schemes achieve better performance with a complexity reduction from $O(n^2)$ to $O(kn/w)$ for n-bit message, k-bit output and w-bit word size.**

***Index Terms*—Algorithm, Cryptography, Hash, MAC, Pseudo-random, Hidden Number Problem**

I. Introduction

The explosive growth of wireless systems coupled with the proliferation of laptop and palmtop computers indicates a bright future for wireless and mobile computing. Smart mobile devices (such as iPad, iPhone, Android, and Blackberry devices) have experimented exponential growth over the last several years and there are currently around 1.2 billion users worldwide [1]. Unfortunately, wireless systems are susceptible to a variety of security attacks due to the openness of the underlying transmission media. Mobile device security has become a major concern to not only mobile clients but also their service providers. To solve this issue, numerous cryptographic techniques (e.g., signature, encryption, identification, etc.) have been developed to provide security protection for these mobile devices. Especially, some basic cryptographic primitives, like Hash, message authentication code (MAC), pseudorandom function (PRF), have been widely employed in a variety of applications and services [1] for various security purposes, including warehouse inventory control, public transportation passes, anti-counterfeiting tags for medicines, secure identification, etc.

In general, these cryptographic primitives are computed frequently on inputs which are usually thousands of bytes long. Computation is typically done through software implementation on relatively weak platforms with meager resources. Additionally, computation is often required to be performed in real time. Therefore, developing optimization techniques for these cryptographic primitives while retaining the appropriate level of security is crucial to improve their performance on mobile devices. We notice that computing power on Arithmetic Logic Unit (ALU) of microprocessors of mobile devices is hardly fully utilized when running these primitives since they were designed based on the simplest hardware construction. Hence, runtime overheads of these primitives could be greatly decreased if we have a reasonable and effective design of cryptographic primitives based on the practical construction in ALU hardware, such as parallel processing units or algebra operation units.

More importantly, there exist too many cryptographic algorithms (or standards) that can be used to implement these primitives. For example, general hash algorithms have MD5, RIPEMD-160, SHA-1, and SHA-256. This often leads to a large number of different algorithms stored in the system, which is a waste of limited storage space, and sometime even causes confusion. Therefore, it is necessary to provide a common method to unify a variety of primitives into a family of cryptographic functions, so that we can use a common core algorithm to construct various cryptographic functions. If so, special hardwares can be designed for the core algorithm to achieve optimal performance. In fact, we expect to design an elaborate and efficient core algorithm by taking full advantage of computation power in mobile devices.

In this paper, we analyze the security and performance features of existing Square Hash function [2]. We show several shortcomings of this function. In response to these shortcomings, we present a new hash scheme (called SqHash) based on a specially truncated function (most significant bits, MSB). We further improve the performance of SqHash by using "circular convolution" which makes parallel processing on output bits possible. Similarly, we present a new MAC scheme (called SqMAC) and a new PRF scheme (called SqPRF), as well as their corresponding improved schemes. We also prove that the security of these constructions based on the privacy properties in the hidden number problem and Hard-core predicate of one-way function. We show that the proposed schemes have better performance as their complexity is reduced from $O(n^2)$ to $O(kn/w)$ for n-bit messages, k-bit outputs and w-bit word-size of mobile devices. The proposed schemes as well as their respective performance and security parameters are summarized in Table I.

II. Background and Preliminaries

A. System Model

In this paper, we assume that a wireless device does not have a dedicated cryptographic hardware. In particular, it does not have a true random number generator. We assume that the device only has a short length (16 bits) accumulator and a 1-bit multiplication unit (it can be replaced by 1-bit accumulator), as well as a large

Y. Zhu works in Beijing Key Laboratory of Internet Security Technology and Institute of Computer Science and Technology, Peking University, Beijing, 100080, China. His work was supported by the National Natural Science Foundation of China (Project No. 61170264 and 10990011).

The work of Di Ma was partially supported by the grant from US National Science Foundation (NSF CNS-1153573).

The work of G.-J. Ahn and H. Hu was partially supported by the grants from US National Science Foundation (NSF-IIS-0900970 and NSF-CNS-0831360).

[1] http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats

978-1-4673-0921-9/12/$31.00 ©2012 IEEE

TABLE I
SUMMARY OF OUR PROPOSED SCHEMES.

Name	Equation	Performance	Security
SqHash	$MSB_k((m\|\|IV) *'_c (m\|\|IV))$	$O(n*k)$	Preimage resistance and Collision resistance
SqMAC	$MSB_k((m+K) *'_c (m+K))$	$O(n*k)$	Secret-key privacy on Hidden Number Problem
SqPRF	$\widetilde{LSB}^u_{2l}((x+i+l) *'_c (x+i+l))$	$O(n*2l)$	Psudeorandom on Hard-core unpredictability

enough memory. In addition, in MAC and PRF, we assume that there exists a secure storage unit which can prevent the disclosure of stored information. It is used to store the user's secret key and the current value of counter.

B. Preliminaries

We refer to $\{0,1\}^*$ as strings and $\{0,1\}^l$ as l-length strings. If x is a string or a number then $|x|$ denotes its length in bits. For $l \in \mathbb{N}$, we denote by 1^l the string of l "1" bits. Let $||$ denote concatenation. For sets X and Y, if $f : X \to Y$ is a function, then we call X the domain, Y the range, and the set $\{f(x)|x \in X\}$ the image of the function. An adversary is an algorithm. By convention, all algorithms are required to be efficient, meaning run in (expected) polynomial-time in the length of their inputs, and their running-time includes that of any overlying experiment.

- **Integer Factoring Problem.** Integer factorization is the decomposition of a composite number into smaller non-trivial divisors. That is, given $N = pq$, for any polynomial time (in $|N|$), and algorithm $\mathcal{A}$ and any polynomial $P(\cdot)$, for sufficiently large $|N|$, the factoring assumption holds that $\Pr[(p,q) = \mathcal{A}] \leq 1/P(|N|)$, where $1/P(|N|)$ denotes a negligible probability and $|N| = 1024$ is often chosen in practice.
- **Square Root Problem.** Let $N = pq$ be the public key and $p \equiv q \equiv 3 \pmod 4$. The Rabin function computes $y = x^2 \pmod N$ for $x \in \mathbb{Z}_N^*$. The square root problem says that given $y \in \mathbb{Z}_N$, without (p,q), to find x such that $y = x^2 \pmod N$ is as hard as factoring N. Note that there are four distinct roots x, including two group trivial roots $(\pm x)$.
- **Hidden Number Problem.** The goal of square hidden number problem (SqHNP) [3] is to find a hidden number s, when given N and access to an oracle that on query $(x_1, \cdots, x_m)$ returns a value $(MSB_k((x+s)^2 \pmod N), \cdots, MSB_k((x_n+s)^2 \pmod N))$, where $MSB_k(y)$ denotes the k most significant bits of y. The SqHNP assumption states that there is no polynomial time algorithm for this problem whenever $k = |N|/3$.

Given a prime $p \equiv 3 \pmod 4$, the square roots of y modulo p can be computed by $x_p = \pm y^{(p+1)/4} \pmod p$. Note that there is no square root for some numbers. When x_p and x_q are known, the square root problem can be resolved by Chinese remainder theorem. It has been proved that decoding the Rabin cryptosystem is equivalent to the integer factorization problem. The Robin cryptosystem is provably secure (in a strong sense) against chosen plaintext attacks.

III. CONSTRUCTION OF HASH FUNCTION

A. Square Hash Function and Its Shortcomings

A cryptographic hash function $Hash : \{0,1\}^* \to \{0,1\}^k$ is a function that takes an arbitrary block of data and returns a fixed-size bit string. A square hash function [2] is a cryptographic hash function $\mathbb{Z}_N \to \mathbb{Z}_N$ for an enough large N, where it is hard to factorize N.

Definition 1 (Basic Construction): The SQH family of hash functions from $\mathbb{Z}_N$ to $\mathbb{Z}_N$ is defined as: $\{f : \mathbb{Z}_N \to \mathbb{Z}_N\}$ where the functions f are defined as

$$f(m) = m^2 \pmod N.$$

Where, $N - N^{1/2} > m > N^{1/2}$ and $|N| = n$.

If $m < 2^{n/2}$, we have $m^2 < 2^n < N$, which means that $m^2 \pmod N = m^2$. In this case, given a hash value $y \in \mathbb{Z}_N$, we can use the split half method (or logarithmic search) to find m such that $m^2 = y$ because m^2 is a monotone increasing function. Similarly, another trivial root of $m^2 \pmod N$ is $N - m < N^{1/2}$ due to $(N-m)^2 = m^2 < N$.

This basic construction has the following shortcomings:

1) Assume that the length of N is n, the computation overhead of SQH function are $O(n^2)$. When n is large, the factorization of N is hard. But $O(n^2)$ is too large for frequent operations in mobile derives.
2) The output of SQH function is $|N|$ bits. Since we use SQH function with at less 1024-bits modulus size for secure integer factorization problem, the hash output size is at less 1024 bits. But the output of SHA-512/384 is just 512 or 384 bits with 1024-bit block size. Hence, the output of SQH function is too large for applications. Also, this feature is contradictory to the compression property of hash functions.
3) When m is too small (such as $m < N^{1/2}$), the output of SQH function is predictable in terms of above discussion. This predictability is considered as one such possible vulnerability that may be insecure for some applications.

Hence, we will focus on addressing these shortcomings by constructing more effective schemes in the remaining of this section.

B. Hash Function for $\mathbb{Z}_N \to \{0,1\}^k$

We present a new construction of square hash function to resolve the shortcomings in the basic construction.

Definition 2 (Square Hash Construction): Define a family of SqHash functions from $\mathbb{Z}_N$ to $\{0,1\}^k$ as: $SqHash = \{f_{IV} : \mathbb{Z}_N \to \{0,1\}^k | IV \in \{0,1\}^l\}$ where the function f is defined as

$$f_{IV}(m) = MSB_k((m||IV)^2 \pmod N).$$

Where, IV denotes an initialization vector and $MSB_k(t)$ denotes k most significant bits of an integer t.

Compared with the basic construction, SqHash has several advantages. Firstly, the output of SqHash is k bits in that $MSB_k()$ can be considered as a truncation function. For instance, k is 384 bits like SHA384. Next, let $m||IV = m \cdot 2^l + IV$. The initial vector IV increases the randomness of the output of SqHash function as a result of $(m \cdot 2^l + IV)^2 = m^2 \cdot 2^{2l} + IV^2 + m \cdot IV \cdot 2^{l+1} \pmod N$, especially for the item $m \cdot IV \cdot 2^{l+1}$. Moreover, this construction also helps avoid collision, that is, given a value y and a square root m ($m^2 = y \pmod N$), we can check IV to determine whether this value is a valid pre-image of the four square roots of y.

Unfortunately, this construction does not reduce the computational complexity because we still need to perform the squaring operation. Hence, we make use of "modulo-$|N|$ circular convolution with carry bit" $m *_c m$ to replace the multiplication $m \cdot m$. The notation $*_c$ for cyclic convolution denotes convolution over

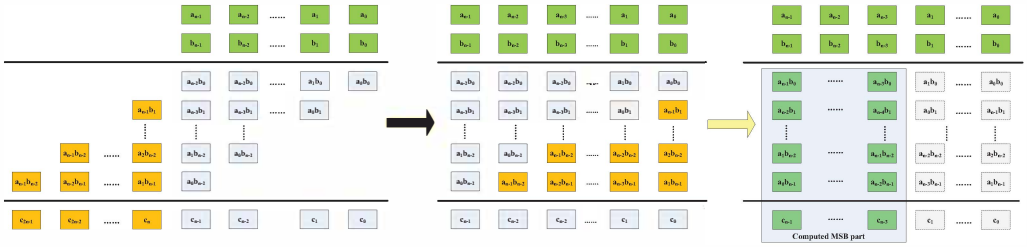

Fig. 1. Multiplication and circular convolution.

the cyclic group of integers modulo $|N|$ [1], that is,

$$(f *_c g)[k] = \sum_{i=0}^{n-1} f[i] \cdot g[(k-i) \mod n],$$

where, $n = |N|$ and $a[k]$ denotes the k-th bit of integer a.

In order to simulate the multiplication operation, we define circular convolution with carry bit as follows:

$$\begin{aligned} r[k] &= (f *_c g)[k] + c_{k-1} \pmod 2 \\ c_k &= ((f *_c g)[k] + c_{k-1} - r[k])/2 \end{aligned}$$

where, $r[k]$ is the k-th output bit, c_k denotes the k-th carry bit, and $c_0 = 0$. In Fig. 1, we show the difference between multiplication operation and circular convolution with carry bit. It is obvious that the latter is a more efficient process. Let $*'_c$ denote above circular convolution with carry bit. The SqHash function can be redefined as

$$f_{IV}(m) = MSB_k((m||IV) *'_c (m||IV)).$$

Note that we do not need module operations in this construction. Furthermore, another advantage of this construction is that we only need to calculate k most important bits without having to calculate other bits. For example, when $N = 1024$ bits and $k = 384$ bits, the computational overheads are $3/8$ of those of multiplication operations. Note that the circular convolution does not change the nature of squaring operation in SqHash scheme. Thus, this transformation from squaring to convolution does not affect the security of SqHash scheme.

C. Hash Function for $\{0,1\}^* \to \{0,1\}^k$

A hash function must be able to process an arbitrary-length message into a fixed-length output. Usually, this can be achieved by breaking the input up into a series of equal-sized blocks, and operating on them in sequence using a one-way compression function. Hence, we define the hash function $\{0,1\}^* \to \{0,1\}^k$ based on the SQH function as follows: given $m = (m_n, \cdots, m_1, m_0)$,

$$\begin{aligned} \sigma_0 &= MSB_k((m_0||IV) *'_c (m_0||IV)); \\ &\cdots \\ \sigma_i &= MSB_k((m_i||\sigma_{i-1}) *'_c (m_i||\sigma_{i-1})); \\ &\cdots \\ \sigma_n &= MSB_k((m_n||\sigma_{n-1}) *'_c (m_n||\sigma_{n-1})). \end{aligned}$$

The final output of hash function is σ_n. The last processed block should also be unambiguously length padded. This is crucial to the security of our construction. Fig. 2 depicts such a construction.

This kind of construction is also called Merkle-Damgärd construction [4], in which any collision for the full hash function can be traced back to a collision in the compression function.

IV. Construction of MAC Function

Message authentication code (MAC), sometimes called keyed (cryptographic) hash function, specifies that an authenticated tag between two parties that share a secret key in order to validate

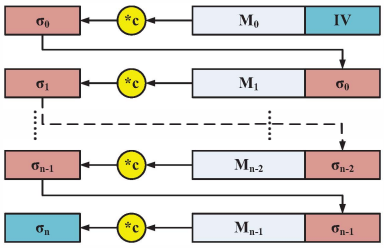

Fig. 2. Hash construction for arbitrary-length messages.

message transmitted between these parties. The MAC value protects a message's data integrity and its authenticity by allowing verifiers (who also possess the secret key) to detect any changes in the message content. Typically, MACs are built from hash functions. Based on square function, we define the square MAC (in short SqMAC) function as follows:

Definition 3 (Square MAC Construction): Let N be an integer to make the factorization attack difficult and $k \in \mathbb{N}$. A square MAC function from $\mathcal{M} \times \mathcal{K}$ to $\{0,1\}^k$ is defined as: $\{SqMAC_k : \mathbb{Z}_N \times \mathbb{Z}_N \to \{0,1\}^k\}$ where

$$SqMAC_k(K, m) = MSB_k((m+K)^2 \pmod N).$$

Where K is a secret key with $|K| = |N|$.

In this construction, $MSB_k()$ is necessary for ensuring security because, without this operation, the secret key K can be computed by

$$K = \frac{\sigma_2 - \sigma_1}{2(m_2 - m_1)} - \frac{m_1 + m_2}{2} \pmod N$$

if we have two pairs (m_1, σ_1) and (m_2, σ_2), where $\sigma_1 = (m_1 + k)^2$ and $\sigma_2 = (m_2 + x)^2 \pmod N$. In addition, we prove that the secret key K cannot be revealed in terms of Theorem 1 even if the adversary has observed a lot of message-MAC pairs.

We also make use of circular convolution to replace the square operation for obtaining higher efficiency, that is,

$$SqMAC_k(K, m) = MSB_k((m+K) *'_c (m+K)).$$

Next, we focus on the MAC construction for arbitrary-length message to be authenticated. HMAC is such a MAC. HMAC(K,m) is mathematically defined by

$$HMAC(K, m) = H((K \oplus opad)||H((K \oplus ipad)||m)),$$

where $H(\cdot)$ denotes a hash function. Based on this construction, we present a new MAC function by extending the function $SqMAC_k(\cdot)$ and above-mentioned hash function for $\{0,1\}^* \to \{0,1\}^k$, as follows:

1) To compute the hash value σ_n based on the algorithm in Section III-C but the initialization vector IV will be replaced by K; and
2) To output the MAC code by using $SqMAC_k(K, m')$ and $m' = K||\sigma_n$.

In this construction, $m' = K||\sigma_n$ is used to expand the length of authenticated message.

V. Construction of Pseudorandom Function

A pseudorandom function, abbreviated as PRF, is an efficiently-computable function which generates a sequence of $\{0,1\}$ that approximates the properties of random string. Strictly speaking, no efficient algorithm can distinguish this sequence from a random sequence with significant advantage. Although there exists some practical PRF algorithms, such as, linear congruential generators (LCG), lagged Fibonacci generators (LFG), and linear feedback shift registers (LFSR), we present a fast PRF construction based on square problem with a simple counter.

A. Secure Square-PRF

In this subsection, we present a new PRF function on the modular square root (MSR) problem. Although we have constructed secure Hash and MAC scheme based on the MSR problem, it is still a challenging task for constructing secure PRF function due to its higher security requirements. Strictly speaking, PRF function can be constructed on the Hard-core predicate of a one-way function, but Hash/MAC function only needs the one-way property [5]. Hence, we first focus on the Hard-core predicate of MSR problem.

To generate pseudorandom number, we assume that a secret x is stored in the device and a counter m_i is used to produce different output of PRF function for each PRF query, that is, $m_{i+1} = m_i + 1$ should be updated automatically. Note that the value m_i should be kept secret. Based on these assumptions, we define a one-way function as

$$f_x(m_i) = (x + m_i)^2 \pmod N.$$

Next, we find out the Hard-core predicate of this function according to the following theorem:

Theorem 1: The two least significant bits (LSB) of next function output $MSB_k(f_x(m_i))$ can only be guessed with $1/3$ probability if m_i is an unknown counter.

Proof: Let $m_{i+1} = m_i + 1$. Given two continuous values $f_x(m_i)$ and $f_x(m_{i+1})$, we have the following equation

$$\begin{aligned} f_x(m_{i+2}) &= (x + m_i + 2)^2 \\ &= (x + m_i)^2 + 4(x + m_i) + 4 \\ &= 2(x + m_i + 1)^2 - (x + m_i)^2 + 2 \\ &= 2f_x(m_{i+1}) - f_x(m_i) + 2. \end{aligned}$$

Assume that e and e' are two $l = |N| - k$ bits numbers. Let $f_x(m_{i+1}) = MSB_k(f_x(m_{i+1}))\cdot 2^l + e \pmod N$ and $f_x(m_i) = MSB_k(f_x(m_i))\cdot 2^l + e' \pmod N$, so that we have the following equation

$$\begin{aligned} MSB_k(f_x(m_{i+2})) &= MSB_k(2f_x(m_{i+1}) - f_x(m_i) + 2) \\ &= 2 \cdot MSB_k(f_x(m_{i+1})) - \\ &\quad MSB_k(f_x(m_i)) + \\ &\quad MSB_k(2e - e' + 2) \pmod N. \end{aligned}$$

In terms of $0 \le e, e' < 2^l$, we have $-2^l < 2e - e' + 2 < 2^{l+1}$. This means that

$$MSB_k(2e - e' + 2) = \begin{cases} -1 & 2e - e' + 2 < 0 \\ 0 & 0 \le 2e - e' + 2 < 2^l \\ +1 & 2^l \le 2e - e' + 2 \end{cases}$$

Hence, $MSB_k(f_x(m_{i+2}))$ has three possible values and each value can be guessed with the same probability $1/3$ according to the difference of $LSB_2(MSB_k(f_x(m_{i+2})))$. ■

This theorem indicates the Hard-core predicate of $f_x(m)$ is $LSB_2(MSB_k(f_x(m)))$. Further, it means that we can guess $2l$-bits in the l-th next value with the probability $\frac{1}{3^l}$. Hence, in order to improve efficiency, the successive $2l$ bits can be outputted $LSB_{2l}(MSB_k(f_x(m_i+l)))$, where m_i is the counter of previous output value. Thus, we define a new square PRF function as follows:

Definition 4 (Secure Square-PRF): Given a counter i, a family of square PRF functions with $2l$-bits output is defined as: $\{SqPRF_{x,i,k} : 1^{2l} \to \{0,1\}^{2l} | x, i \in \mathbb{Z}_N, k \in \mathbb{N}\}$, where the function $SqPRF$ is defined as

$$SqPRF_{x,i,k}(1^{2l}) = LSB_{2l}(MSB_k((x + i * l)^2 \pmod N)).$$

Where, $MSB_k(t)$ and $LSB_k(t)$ denote k most and least significant bits of an integer t, the counter is updated to $i * l$ after this process, and $N - N^{1/2} > x > N^{1/2}$.

B. Efficient Implementation

To implement $SqPRF$ function, we can still use the "circular convolution with carry bit" to reduce the computational overheads, but we must deal with how to compute a particular bit (specially for LSB bits) correctly in $m^2 \pmod N$. More precisely, in order to be certain about the effect of the carry entering this bit (or LSB_l) position, we have to compute all earlier bits in the worst case. Fortunately, it is easy to show that the carry into each bit position in the computation of m^2 can be at most 11 bits long [2] for $|N|$ between 1024 and 2048. Thus, if we add $u = 32$ additional low order bits to the computed window, we have only a small (negligible) probability of less than $2^{11}/2^{32} = 1/2^{21}$ of computing an incorrect carry into the 33-th bit we compute. We call it "least significant bits with window" ($\widetilde{LSB}_l^u$). We present this process in Fig. 3.

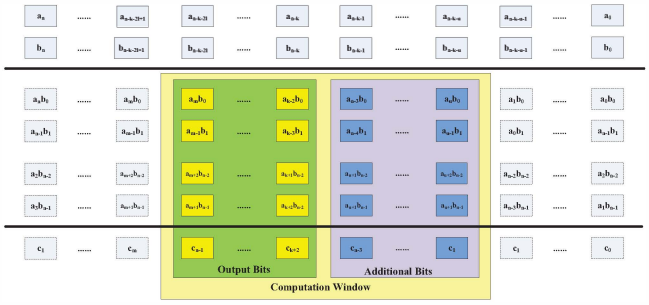

Fig. 3. Circular convolution for least significant bits with window.

So that we can replace the above $SqPRF$ function by using

$$SqPRF_{x,i,k}(1^{2l}) = \widetilde{LSB}_{2l}^u((x + i + l) *_c (x + i + 1)).$$

This can guarantee an extremely small error probability while keeping the average running time only slightly higher than always computing $l + 32$ bits.

VI. Security Analysis

A. Security analysis of SqHash functions

The SQH function $f_{IV}(m)$ is a cryptographic hash function with preimage resistance and collision resistance [6]. The latter property also means that this function is second-preimage resistance. The preimage resistance stems from the fact that Square function $y = x^2 \pmod N$ is a trap-door one-way function in that given y, without (p, q), to find x such that $y = x^2 \pmod N$ is as hard as factoring N.

The collision resistance is based on the infeasibility of integer factoring problem [7]. Assume that given y, there exist two values

[2]The value of carry bit is at most $2^{11} - 1$, so it can effect at most $\log_2(2^{11} - 1) \approx 11$ bits long.

x_1 and x_2 for $(x_1||IV)^2 = (x_2||IV)^2 \pmod N$ and $x_1 \neq x_2$, such that we have $(x_1||IV)^2 - (x_2||IV)^2 = (x_1 - x_2)((x_1 + x_2)2^l + 2 \cdot IV) \cdot 2^l = 0 \pmod N$. This means that $N|(x_1 - x_2)((x_1+x_2)2^{l-1}+IV)$. Thus, we can find a factor of N by using $\gcd((x_1-x_2), N)$ only if not $N|((x_1+x_2)2^{l-1}+IV)$. Otherwise, we can repeat this process to find out the factor of N. Further, given a fragment $MSB_k(y)$ of y, it is harder to find a valid x which includes IV, that is, $y = (x||IV)^2 \pmod N$. Even if with (p, q), it is hard to find a valid e with $y = MSB_k(y)||e$ such that $y = (x||IV)^2 \pmod N$.

B. Security analysis of SqMAC functions

Given the function $SqMAC_k(K, x)$, we prove that the secret key K cannot be revealed even if the adversary obtains sufficient message-MAC pairs. In this proof, we consider a hidden number problem: Let K be a random hidden element of $\mathbb{Z}_N$. We are given $N, k = |N|/3, |N| = n$, and $(x_i, MSB_k((x_i+K)^2 \pmod N))$ for random values $x_1, \cdots, x_t$. The problem is to find K. That is, we assume that $MSB_k((x_i + K)^2 \pmod N)) = y_i$, so that we have

$$x_i^2 + K^2 + 2x_iK = y_i \cdot 2^k + e_i \pmod N, \qquad i = 1, \cdots, t$$

where e_i are variables that correspond to unknown low order bits and $|e_i| \leq 2^{n-k} = 2^{2|N|/3}$. We are therefore forced to eliminate unknown K^2 from the above relation by using the equation:

$$x_i^2 - x_1^2 + 2(x_i - x_1)K = (y_i - y_1)2^k + (e_i - e_1) \pmod N.$$

Next, we also eliminate K by the equation:

$$\begin{aligned}(x_i^2 - x_1^2)(x_i - x_1) - (x_j^2 - x_1^2)(x_j - x_1) \\ = \quad (x_i - x_1)(x_j - x_1)(x_i - x_j) \\ = \quad (x_i - x_1)(y_i - y_1)2^k - (x_j - x_1)(y_j - y_1)2^k + \\ (x_i - x_1)(e_i - e_1) - (x_j - x_1)(e_j - e_1) \pmod N.\end{aligned}$$

Also, we rewrite this equation as a polynomial in the unknown e_i, e_j, e_1, namely:

$$f_i(e_i, e_j, e_1) := A_{i,1}e_i + B_{j,1}e_j + C_{i,j}e_1 - X_{i,j} \pmod N.$$

where $A_i = x_i - x_1$, $B_j = x_1 - x_j$, $C_j = x_j - x_i$, and $X_i = ((x_i - x_1)(y_i - y_1) - (x_j - x_1)(y_j - y_1))2^k - (x_i - x_1)(x_j - x_1)(x_i - x_j)$. Based on this function, we setup a lattice of dimension $2t - 1$ as a real matrix M:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & X_{3,2} & \cdots & X_{t,2} \\ 0 & 2^{k-n} & 0 & 0 & \cdots & 0 & C_{3,2} & \cdots & C_{t,2} \\ 0 & 0 & 2^{k-n} & 0 & \cdots & 0 & B_{2,1} & \cdots & B_{2,1} \\ 0 & 0 & 0 & 2^{k-n} & \cdots & 0 & A_{3,1} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 2^{k-n} & 0 & \cdots & A_{t,1} \\ 0 & 0 & 0 & 0 & \cdots & 0 & N & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & N \end{pmatrix}$$

Next, we set $v = (1, e_1, e_2, \cdots, e_t, k_2, \cdots, k_t)$. It follows that for this integer vector v we get:

$$v \cdot M = (1, \frac{e_1}{2^{n-k}}, \cdots, \frac{e_t}{2^{n-k}}, 0, \cdots, 0).$$

Thus, the lattice point $v \cdot M$ has only $t + 1$ non-zero entries, and each of these is less than 1. And its Euclidean norm is less than $\sqrt{t+1}$. On the other hand, it is easy to see that the determinant of the lattice $L(M)$ equals $N^{t-2} \cdot 2^{(k-n)t}$. In addition, making using of the Gaussian heuristic for short lattices vectors, we expect that our vector is the shortest point in this lattice $L(M)$ as long as $\sqrt{t+1} \ll \sqrt{2t-1}(N^{t-2} \cdot 2^{(k-n)t})^{1/(2t-1)}$. Let $N \approx 2^n$ and ignoring low-order terms, this condition is simplified to $\sqrt{\frac{t+1}{2t-1}} \approx 2^{-1/2} \ll 2^{k/2-n/t} < 2^{k/2} = 2^{n/6}$. Therefore, the Lattice-based reduction methods cannot find the secret K [8].

C. Security analysis for SqPRF function

The security requirement for PRF function is that the output of pseudorandom sequences should be computationally indistinguishable from truly random sequences. For practicality, such pseudorandom generators can be constructed easily on the hard-core predicates. In fact, we have proved that our SqPRF function is constructed on the hard-core predicates with the probability $1/3^l$ for $2l$-bits outputs in Theorem 1.

VII. Performance of Algorithms

We use two different approaches (i.e., the truncated function and circular convolution) to improve the performance of all proposed schemes. To evaluate the performance, we use Mathematica to realize our schemes ($|N| = 1024$ and $k = 384$) and some standard hash functions, such as, MD5, SHA1, SHA384, SHA512. We find that our SqHash, SqMAC, SqPRF schemes and these standards have almost comparable overheads.

Usually, long-size cryptographic algorithms are not suitable for providing security on wireless devices due to their limited computation and communication capabilities. In a cryptosystem, key length is usually much longer than the "word size" of mobile devices, typically 16 or 32 bits. Using short ALU to deal with long data often leads to more complex programs for cryptography algorithms. Moreover, traditional Hash/MAC/PRF functions do not contain algebra operations thus ALU computing power would normally not be fully utilized. However, the algorithms proposed in this paper are easy to implement, and can take advantage of ALU computing power. We will give basic construction on MSB and convolution, on which SqHash/SqMAC/SqPRF can be easily constructed.

First, our basic construction (called "Convol-MSB" algorithm in Fig.4) can be implemented easily on mobile devices. We improve the computation by replacing integer multiplication with circular convolution. The advantage of this method is that each output bit is calculated from right to left, while traditional multiplication is from top to bottom, then right to left (see Fig. 1). In addition, we also use MSB operation (with window) to further reduce computation overheads. Based on them, the computation complexity of "Convol-MSB" is $O(kn)$ instead of $O(n^2)$ in the traditional multiplication way. It is about k/n of the original overhead.

The basic "Convol-MSB" algorithm is constructed on bit-AND operator. Thus, it does not take full advantage of the ALU computing resources. Taking into account of bit-parallel processors in ALU hardware, we improve the above-mentioned algorithm to a new "Convol-MSB-IMP" algorithm in Fig. 5. In this algorithm, we make use of $CyclicLeftShift(x, k)$ to cycles k positions to the left for the elements in x. Furthermore, we use $a*.b$ to give the bitwise AND of a and b, and $Sum(x)$ to give the total number of elements in list x. Based on these improvements, the computation overhead is further reduced to $O(kn/w)$ since the actual scale is the word size of mobile devices w in each parallel processing. Note that above pseudo-codes are not real ALU calls and we need to expand these calls according to actual input length.

TABLE II
COMPARISONS AMONG SQHASH, SQMAC, SQPRF AND THEIR IMPROVED SCHEMES.

Name	Equation	Performance	Improved Equation	Performance
SqHash	$MSB_k((m\|\|IV)^2 \pmod N)$	$O(n^2)$	$MSB_k((m\|\|IV) *'_c (m\|\|IV))$	$O(n*k)$
SqMAC	$MSB_k((m+K)^2 \pmod N)$	$O(n^2)$	$MSB_k((m+K) *'_c (m+K))$	$O(n*k)$
SqPRF	$LSB_{2l}(MSB_k((x+i+l)^2 \pmod N))$	$O(n^2)$	$\widetilde{LSB}^u_{2l}((x+i+l) *'_c (x+i+l))$	$O(n*(2l+u))$

Algorithm Convol-MSB(x, n, k)
Require: x is an input integer, n is the length of x, and k is the length of output result.
$s = 0$;
for $i = n - k + 1$ to n **do**
  for $j = 1$ to n **do**
    $r = x[j] \cdot x[(i-j) \pmod n]$;
    $s = s + r$;
  end for
  $idx = i - n + k$;
  $r[idx] = s \pmod 2$;
  $s = (s - r[idx])/2$;
end for
Return r;

Fig. 4. Basic convolution-MSB algorithm.

Algorithm Convol-MSB-IMP(x, n, k)
Require: x is an input integer, n is the length of x, and k is the length of output result.
$t_x = \text{CyclicLeftShift}(x, n-k)$;
$s = 0$;
for $i = 1$ to k **do**
  $t_x = \text{CyclicLeftShift}(x, 1)$;
  $t = x * .t_x$;
  $r[i] = sum(t+s) \pmod 2$;
  $s = (sum(t+s) - r[i])/2$;
end for
Return r;

Fig. 5. Improved convolution-MSB algorithm.

In Table II, we give a summary of the performance estimate of SqHash, SqMAC, and SqPRF compared with corresponding improved schemes. Obviously, the improved scheme achieves a computational overhead ($O(nk)$) smaller than the original schemes ($O(n^2)$). The ratio between them is about $k/n = O(nk)/O(n^2)$. This result is benefited from the use of truncated function MSB or LSB, as well as circular convolution operation. In addition, these truncated functions also increase the difficulty of attacks against the improved schemes. Moreover, the overhead $O(nk)$ means that the scheme is less sensitive than the original scheme for data length n. For example, when we want to double the length of processed message, the overhead of improved schemes will increase by about 100% (from $O(nk)$ to $O(2nk)$), but those of original schemes increase by about 400% (from $O(n^2)$ to $O(4n^2)$).

To validate the efficiency of our approach, we implemented several algorithms in the Mathematica 7.0 environment. In Fig. 6, we show a comparison of experimental results among Square, convolution-MSB and improved convolution-MSB algorithms. In this figure, the total length of N is changed from 512 to 1024-bits and the size of output results is about 1/3 of length of N. The computation cost is proportional to $|N|$ for the square algorithm and the convolution-MSB algorithm and the length of N has a greater impact on the Square algorithm. However, the computation cost

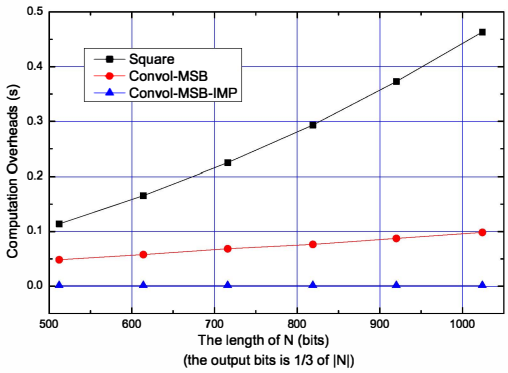

Fig. 6. Comparison of experimental results among Square, convolution-MSB and improved convolution-MSB algorithms.

is independent to the length of N in the improved convolution-MSB algorithm. In summary, the experimental results show that our improved convolution-MSB method has better performance than general square algorithm.

VIII. CONCLUSIONS

In this paper, we prompt the idea of constructing various basic cryptographic primitives (such as hash, MAC, and PRF) from a common core algorithm to simplify the management and improve the efficiency of current cryptographic implementation practices on mobile devices. For this purpose, we present the design of a family of cryptographic primitives based on the common core squaring operation. Our design takes advantage of the practical construction in ALU hardware, such as parallel processing units or algebra operation units, so that the proposed schemes can be efficiently realized on resource-constrained mobile devices. Moreover, the proposed schemes are provably secure under the hidden number problem and hard-core predicate.

REFERENCES

[1] A. Shamir, "SQUASH - a new MAC with provable security properties for highly constrained devices such as RFID tags," in *FSE*, 2008.
[2] M. Etzel, S. Patel, and Z. Ramzan, "Square hash: Fast message authenication via optimized universal hash functions," in *CRYPTO*, 1999.
[3] D. Boneh, S. Halevi, and N. Howgrave-Graham, "The modular inversion hidden number problem," in *ASIACRYPT*, 2001.
[4] J.-S. Coron, Y. Dodis, C. Malinaud, and P. Puniya, "Merkle-damgård revisited: How to construct a hash function," in *CRYPTO*, 2005.
[5] C. Berbain, H. Gilbert, and J. Patarin, "Quad: A practical stream cipher with provable security," in *EUROCRYPT*, ser. Lecture Notes in Computer Science, S. Vaudenay, Ed., vol. 4004. Springer, 2006, pp. 109–128.
[6] K. Ouafi and S. Vaudenay, "Smashing squash-0," in *EUROCRYPT*, 2009, pp. 300–312.
[7] V. Dubois, P.-A. Fouque, A. Shamir, and J. Stern, "Practical cryptanalysis of sflash," in *CRYPTO*, ser. Lecture Notes in Computer Science, A. Menezes, Ed., vol. 4622. Springer, 2007, pp. 1–12.
[8] D. Boneh and R. Venkatesan, "Rounding in lattices and its cryptographic applications," in *SODA*, M. E. Saks, Ed. ACM/SIAM, 1997, pp. 675–681.