

SQUASH: A Provably Secure One-way Hash Function for Highly Constrained Devices such as RFID Tags

**Adi Shamir
Computer Science Department
The Weizmann Institute of Science
Israel**

1

10/25/2007

Passive RFID tags:



10/25/2007

2

Passive RFID tags have lots of potential applications:

- warehouse inventory control
- supermarket checkout counters
- public transportation passes
- anti-counterfeiting tags for medicines
- pet identification
- Secure passports
- . . .

10/25/2007

3

There are two very different types of RFID applications:

- **Low cost applications**, in which functionality is important and security is a nice to have extra feature
- **High security applications**, in which security is crucial and other features are a secondary consideration

10/25/2007

4

THE MINIMALIST APPROACH TO LOW COST RFID SECURITY:



The main problems in the world of low cost passive RFID tags:

- Cost per device
- Operational range

consequently, everything is severely limited:

- **Number of gates in the controller** (typically 1K to 10K, with 1/3 devoted for security)
- **Size and type of memory** (several hundred bits, only some of them in nonvolatile rewritable memory)
- **Processing speed** (to simplify the design and to reduce power consumption)
- **Bandwidth** (to some extent)
- ...

10/25/2007

7

Security implications:

- The device should use a **small number of small cryptographic keys and/or short passwords**. Storing even a single 1024 bit RSA key can be a severe problem
- Most of the standard cryptographic functions (including **SHA-1, AES, RSA, ECC**, etc) are completely inappropriate for low cost RFID's

10/25/2007

8

*I have a strong feeling of **déjà vu**:*

- We had similar problems in the early days of smart cards, in the late 1980's

10/25/2007

9

*Let us concentrate on the most important RFID security functionality: **Authentication***

- The RFID knows a short secret value **S**.
- It wants to convince the real reader that it **knows S**.
- ZK proofs are an overkill, since we assume that the honest reader **already knows S**, and can use his knowledge during verification.
- The goal of the protocol is to protect the secrecy of **S** against **passive eavesdroppers**, and to prevent replay and other attacks **later** by **active adversaries**.

10/25/2007

10

The standard solution: Challenge and response using a strong hash function

- A hash function H is known to everyone
- A secret key S is shared by RFID and reader
- The reader sends a random challenge R
- The verifier sends $H(S, R)$ to the reader
- The reader **re-computes locally** the expected value, and **compares the result** with the message received from the RFID
- Simple extensions allow secure two-way authentication by the RFID and reader

10/25/2007

11

What are the security requirements?

- The adversary should **not be able to answer a new random challenge R** even after seeing the answers to many (known or chosen) other challenges R_i .
- In particular, **S itself should remain secret.**
- We do not actually care about **collision resistance**, which is the main design consideration for hash functions.

10/25/2007

12

What are the *optimization criteria*?

- H should have short inputs and outputs
- H should use very few small internal registers
- H should have very simple description
- H should consume very little power
- H should be reasonably fast, and executed only a small number of times in the protocol

10/25/2007

13

Which hash function should we use?

- Standard hash functions such as SHA-1 and SHA-256 are a security overkill and performance hogs in this application
- We need hash functions which emphasize only the noninvertibility issue and which are easier to implement on passive RFID tags.
- It is easy to come up with very simple hash functions, but we would like to have some provable security properties

10/25/2007

14

The best solutions proposed so far: The HB family started by Hopper&Blum

- The original HB protocol
- The HB+ protocol
- The HB++ protocol
- The HB-MP protocol

10/25/2007

15

The original HB protocol (Hopper, Blum):

- Both parties know n secret bits
 x_1, x_2, \dots, x_n
- Verifier sends random challenge
 a_1, a_2, \dots, a_n
- Prover computes $A = a_1x_1 + \dots + a_nx_n$
- Prover sends randomized answer
 $H = A + r$

10/25/2007

16

The improved HB+ protocol (Juels, Weis):

- Both parties know $2n$ secret bits x_1, x_2, \dots, x_n and y_1, y_2, \dots, y_n
- Verifier sends random a_1, a_2, \dots, a_n
- Prover chooses random b_1, b_2, \dots, b_n
- Prover computes $A = a_1x_1 + \dots + a_nx_n + b_1y_1 + \dots + b_ny_n$
- Prover sends randomized answer $H = A + r$ and b_1, \dots, b_n to verifier

10/25/2007

17

Man in the middle attack on the HB+ protocol (Gilbert, Robshaw, Sibert):

- Attacker consistently modifies all the verifier's challenges into $a'_i = a_i + c_i$
- Attacker passes the prover's answers
- Attacker watches the verifier's reaction
- If the verifier accepts the proof, attacker concludes that $c_1x_1 + \dots + c_nx_n = 0$
- Attacker repeats the attack n times with different c_i , and solves the linear equations

10/25/2007

18

Problems with these solutions:

- H has to be evaluated many times
- Tag has to generate many random bits
- Probabilistic proof may be rejected by reader
- Many attacks and countermeasures proposed
- Uncertainty in the recommended parameters

10/25/2007

19

My crazy recommendation:

- Use a standard and well studied **public key scheme** as a hash function!

10/25/2007

20

An earlier attempt to use modular exponentiation as a (very slow) hash function:

$$H = e^m \pmod{n = pq}$$

- The goal was to achieve **provable collision resistance** if factoring is difficult
- Any pair of messages m_1 and m_2 that collide have a difference which is a multiple of $\phi(n)$, whose knowledge is **provably equivalent to factoring n** .
- Completely unsuitable for RFID's

10/25/2007

21

Let us consider the Rabin variant of the RSA function:

$$c = m^2 \pmod{n}$$

- This mapping is believed to be a **very good one-way function**
- It had been studied for **more than 30 years**
- Finding m (or some of its bits) from c is **provably as difficult as factoring n** .
- This is true for **each n separately**, regardless of its form

10/25/2007

22

The Rabin scheme seems to be a very bad candidate for RFID's:

- It requires large registers
- It produces large outputs
- It uses a lot of power
- It is too slow

10/25/2007

23

Note that the standard Rabin mapping is **not collision resistant**:

- Modular squaring is a very bad hash function in the classical sense, since it is very easy to create collisions:
 - $m^2 \pmod n = (-m)^2 \pmod n$
 - $m^2 \pmod n = (m+n)^2 \pmod n$

10/25/2007

24

The size of the code for Rabin's scheme:

- The good news: Non-modular multiplication is highly regular, and can be written with **a tiny piece of code** with 1-bit multiply/add instructions and two nested loops (much smaller than AES or SHA-1!)
- The bad news: a modular multiplication is **considerably more complicated**

10/25/2007

25

Flashback: The Randomized Rabin Scheme

- I already showed several years ago how to eliminate modular reductions in a way which is **provably equivalent to modular squaring**
- That scheme had **full public key functionality**.
- In this presentation I will **further simplify the Rabin scheme when we do not need that functionality**.

10/25/2007

26

How to eliminate the modular reduction in the randomized Rabin scheme:

$$c = m^2 \pmod{n}$$

$$c = m^2 - an$$

(for some carefully computed coefficient a which is very difficult to derive with small memory)

10/25/2007

27

Use randomized squaring instead of modular reductions:

The randomized squaring of m is computed as:

$$c = m^2 + rn$$

for a randomly chosen r which is slightly larger than n

The result is computationally equivalent to the original Rabin scheme, since anyone can later reduce this value mod n .

A disadvantage is that this result is twice as long as the standard result of modular multiplication, but bandwidth is not a severe limitation in our application.

10/25/2007

28

Sketch of the security proof:

Given

$$c = m^2 \pmod{n}$$

Anyone can compute the distribution

$$m^2 - an + rn = m^2 + (r - a)n$$

Question: Is it distinguishable from the distribution of $m^2 + rn$

10/25/2007

29

Picture proof:



Randomly sampling the two combs is very unlikely to expose their difference

10/25/2007

30

Our game plan:

- **Optimize, optimize, optimize:** simplify the Rabin scheme to its barest minimum when we do not want to invert this function, do not need its public key functionality, and do not care about its collision resistance
- **Play the blame game:** If my scheme is broken, its not me, its Rabin who should be blamed...

10/25/2007

31

Our solution in a nutshell:

- We propose to compute an excellent numerical approximation to a window of t consecutive bits in the middle of $m^2 \pmod n$ where n has special form.
- We call the new hash function **SQUASH** which is a squashed form of **SQUare-hASH**
- We will have to do two things:
 - Show how to compute this window of bits much more efficiently than the full Rabin result
 - Prove that SQUASH is at least as secure as the full Rabin scheme in the challenge/response application

10/25/2007

32

Idea 1: Don't store m

- In the Rabin scheme, the modulus n should have 1000+ bits, and the message m should be full size
- To generate m , XOR the 64-bit challenge R into the 64-bit secret S and the short challenge R , and use it as a seed for a reversible non-linear feedback shift register (NLFSR)
- This makes it easy to generate successive bits of m on the fly in the forward or backward direction:
 $m = \text{NLFSR}(S \text{ XOR } R)$

10/25/2007

33

Idea 2: Use only a short window of bits in the Rabin ciphertext as output of H:

- We need $t=32$ to 64 bits of output, not 1000+
- Can this make SQUASH **weaker** than Rabin?
- There are **two types of attacks** on SQUASH:
 - Provide the correct answer for a new R without having any useful information about S
 - Exploit some useful information about S extracted from analyzing previous answers

10/25/2007

34

Providing correct answers when no useful information on S is known:

- The Rabin mapping is **almost a permutation** and thus every value of the window occurs about the same number of times for all m
- When m is expanded from short S and R, we have to make **mild additional assumption on the expansion**
- Given a new challenge R, the first type of attack succeeds with probability of about 2^{-t}

10/25/2007

35

Extracting useful information on S:

- The difficulty of extracting information on S is **monotonically decreasing** with the size t of the available window: If a predicate of S can be computed from such windows, it can also be computed from the full Rabin answers.

10/25/2007

36

Remarks:

- When we use the **standard Rabin scheme**, the correctness of such a window can be easily verified, but its computation is not much faster than full Rabin
- When we use the **randomized Rabin scheme**, such windows can be computed much faster, but their correctness cannot be easily verified by the reader

10/25/2007

37

Idea 3: To simplify deployment, use a universal n

- *Since no one has to invert H , everyone can use **the same universal n** if no one knows how to factor it.*
- *This idea was used before (e.g., in the Fiat-Shamir identification scheme).*
- *One way to implement it is to choose p and q , multiply them, print the result, and then **blow up the computer**.*

10/25/2007

38

Idea 4: Don't store n

- It is easy to find $n=pq$ whose **top half** is any desired value, such as 100000...
- It is also possible to choose a **large pseudo random n** without knowing its factorization, and to check that it has no small factors
- The **Cunningham project numbers** $n=a^b+-c$ for small a,b,c are **highly compressible numbers** which were **extensively tested** by a large community.

10/25/2007

39

Which size should we use?

- Such numbers are **somewhat easier** to factor (using the special number field sieve) than general RSA numbers (using the general number field sieve)
- We have to make them somewhat larger
- My recommendation: use **1200-1600 bit moduli**

10/25/2007

40

Example: Factorizations of 2^k-1

■	1201	57649.1967239.8510287.2830858618432184648159211485423.	C314
■	1213	327511. C360	
■	1217	1045741327. C358	
■	1223	2447.31799.439191833149903. P346	
■	1229	36871.46703.10543179280661916121033. C339	
■	1231	531793.5684759.18207494497.63919078363121681207. C329	
■	1237	C373	
■	1249	97423.52358081.2379005273.9345276045907272726364012481. C326	
■	1259	875965965904153. C365	
■	1277	C385	
■	1279	P386	
■	1283	4824675346114250541198242904214396192319. C347	
■	1289		
		15856636079.108817410937.827446666316953.9580889333063599.1605582695344	
		8199975207. P314	
■	1291	998943080897.84051400422953302189544581841. C348	
■	1297	12097392013313.64873964199444497. C361	
■	1301	161317830296866767945829203107381353. C357	

10/25/2007

41

My recommendation: Use the following **Mersenne numbers 2^k-1**

- $2^{1277}-1$ is a 160 byte / 385 decimal digit composite number with no known factors
- Alternatives: $2^{1237}-1$ or even $2^{1061}-1$

10/25/2007

42

Remarks:

- *Unlike the case of the original Rabin scheme, a future complete factorization of such a universal modulus n will have **only limited impact** on SQUASH*
- *It will eliminate the **formal proof of security***
- *There is no known way how to use such a factorization to **actually extract S** from short windows of bits in the middle of $m^2 \pmod{n}$*

10/25/2007

43

Remarks:

- *We will later prove that we can use moduli **with some small known factors** without changing anything in the hashing scheme and without losing any security*
- *It is easy to use **other numbers in the Cunningham tables***

10/25/2007

44

Idea 5: compute the multiplications by on-the-fly convolutions

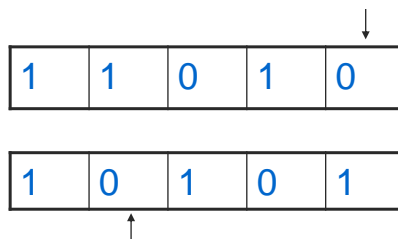
- The RFID will compute the bits of the middle window in c from **LSB to MSB**.
- Each bit will be **sent out as soon as it is computed**, and erased from memory.
- The only memory we need is an **12-bit register** for the carry generated by the previous steps in the convolution.

10/25/2007

45

Computing the multiplications:

- To compute the non-modular $m \times m$, use the standard technique of **successive convolutions**, which requires two 11 bit pointers and a 12 bit adder:



The previous carry and current sum:



Idea 6: Approximating rather than computing the value of the window

- **Problem:** How do we know **the first carry into the LSB of the window?**
- If we want to know it **for sure**, we have to compute all the previous bits. If we only want to **approximate it**, this is much easier
- **Solution:** **extend the lower end** of the k -bit window with u additional bits.
- Compute the value of the extended window **assuming that the carry into its LSB was zero**
- At the end of the computation of the $k+u$ bits of the extended window, **throw away the lower u , and provide the top k as the output of H**

10/25/2007

47

Why the security remains unaffected

- If u is sufficiently large (e.g., $u > 64$), the approximation is so good that an attacker will never see a difference between the approximate and real values in the Rabin scheme
- Consequently, if the scheme that computes the real value of a Rabin window is secure, the scheme that computes an indistinguishable distribution will also be secure.

10/25/2007

48

Efficient computation of the good approximation:

- Adding 64 guard bits to a 32 bit result will be *expensive and unnecessary*
- Adding only 16 guard bits may be *insufficient*
- Since the carry is at most 11-bit long, *we can know* when 16 guard bits might be insufficient, and recompute with a larger number of guard bits only in a small percentage of cases, which has small impact on the average running time

10/25/2007

49

How to use Mersenne numbers with some small known factors:

- Information about m modulo the product W of all the small known primes *will leak out* in this case
- This can be *completely avoided* if we randomize c by adding to it a random value E between 0 and W .
- Note that the addition of the small E is extremely unlikely to affect the middle window, so we can *pretend that it was added without doing anything!*

10/25/2007

50

Idea 7: Modular reduction mod $n=2^k-1$

- When $n=2^k-1$ the computation of $m^2 \pmod n$ is particularly simple since $2^k=1 \pmod n$:



Top half:



Bottom half:



10/25/2007

51

To compute bit j in lower half:

- Sum (over the integers, not modulo 2) all the products $m_v * m_{j-v}$ for $v=0, 1, 2, \dots, j$, and add to this sum the carry from the computation of the previous bit.

10/25/2007

52

To compute bit $k+j$ in upper half:

- *Sum (over the integers, not modulo 2) all the products $m_v * m_{j+k-v}$ for $v=j+1, \dots, k-1$, and add to this sum the carry from the computation of the previous bit*

10/25/2007

53

To compute bit j in $c=m^2 \pmod{2^k-1}$:

- *Add bits j and $j+k$ in m^2 , along with their carries.*

10/25/2007

54

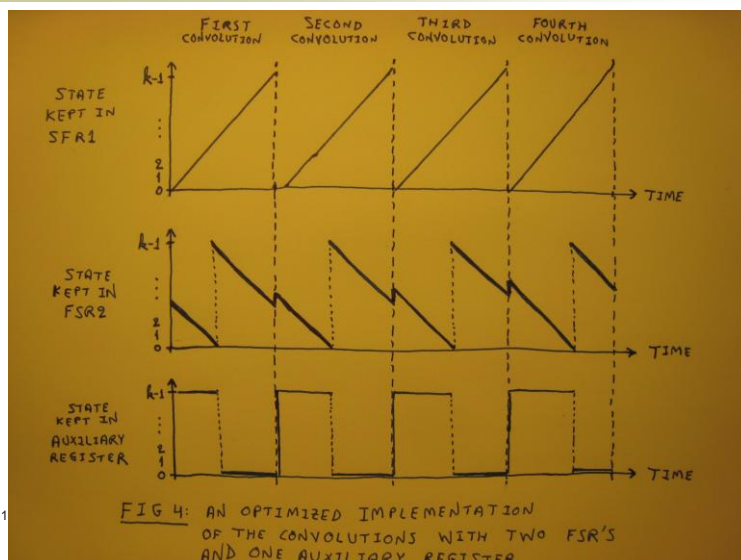
The final SQUASH algorithm is extremely simple:

- Set $j=600$ and set $\text{carry}=0$.
- Compute $\text{carry}=\text{carry}+m_v * m_{j-v \pmod k}$ for $v=0, 1, \dots, 1276$ (using integer addition).
- Set $c_j=\text{LSB}(\text{carry})$, $\text{carry}=\text{rightshift}(\text{carry})$.
- Repeat steps 2 and 3 48 times, and output the 32 bits c_{616}, \dots, c_{647}

10/25/2007

55

Efficient implementation with two FSR's and an auxiliary register:



56

How to exchange the values of two registers without using extra storage:

- To exchange the values of Y and Z:
- XOR Z into Y
- XOR Y into Z
- XOR Z into Y

10/25/2007

57

Remarks:

- The generation and the verification are so easy that **both of them** can be implemented on RFID's or cheap sensor networks
- On a more powerful machine such as a PC with 32-bit multipliers, the evaluation of SQUASH will be **extremely fast**.
- The computation can be **easily parallelized**, and we can use Feistel structures rather than FSR's to generate larger chunks of m in each clock cycle

10/25/2007

58

Connections with other schemes:

- *HB/HB+*: Compute a dot product of a *secret vector* with a *known vector*
- *QUAD*: Compute *dense quadratic maps* of the input bits

10/25/2007

59

Conclusions:

- *In this talk I presented very simple and practical solutions to the problem of authenticating low cost passive RFID's*
- *These solutions should be tested with actual implementations*
- *There are many possible extensions and variations which should be investigated further*

10/25/2007

60