The Bit-Flip Protocol

Verifying A Client With Only Near Zero Computing Power:

Protecting IOT Devices from Serving the Wrong Client

Gideon Samid
Department of Electrical Engineering and Computer Science
Case Western Reserve University, Cleveland, OH
BitMint, LLC
Gideon@BitMint.com

Abstract: The majority of IOT devices have near zero computing power. They respond to wireless commands which can easily be hacked unless encrypted. Robust encryption today requires computing power that many of those sensors that read temperatures, humidity, flow rates, or record audio and video -- simply don't have. The matching actuators that redirect cameras, open/close pipelines etc. – likewise, don't have the minimum required computing capacity, nor the battery power to crunch loaded number-theoretic algorithms. We propose a solution where the algorithmic complexity of modern cryptography is replaced with simple *bit-wise primitives*, *and where security is generated through large (secret) quantities of randomness*. Flash memory and similar technologies make it very feasible to arm even the simplest IOT devices with megabytes, even gigabytes of high quality randomness. We propose to exploit this high quantity of randomness to offer the required security, which is credibly assessed on the sound principles of combinatorics. For example: a prover will send a verifier their shared secret S, after flipping *exactly half of S bits*. For any third party the flipped-bits string will be comprised of bits such that each bit has 50% chance to be what it is, or to be the opposite. For the verifier the risk that the communicator of the flipped-bits string is not in possession of the shared secret S is (i) very well established via combinatoric calculus, and (ii) is getting smaller for larger strings (e.g for |S|=1000 bits, there is 2.5% chance for a fraud, and by repeating the dialogue, say 4 times the risk if less than 1 in a million).

Table of Contents

Introduction

The magic of global access offered by the Internet, is about to be extended ten fold to 60 or 70 billion devices sharing a cyber neighborhood. The promise of the Internet of Things is mind boggling, but on second glance one wonders if the ills of cyber wrongs and cyber criminality will not also multiply ten fold. We envision a world where billions of sensors read their environment, and billions of actuators control and manipulate the same environment -- all for our benefit. But alas, with so much that is done by the IOT to support our modern life, there is so much of a risk of abuse and malpractice to mis-apply the same. Recently some researchers warned about the "nuclear option" where compact clusters of IOT devices will spread malware in an "explosive" uncontrollable way [Ronen 2016]. The same authors warn: "*We show that without giving it much thought, we are going to populate our homes, offices, and neighborhoods with a dense network of billions of tiny transmitters and receivers that have ad-hoc networking capabilities. These IoT devices can directly talk to each other, creating a new unintended communication medium that completely bypasses the traditional forms of communication such as telephony and the Internet".*

In the "old Internet" we build integrity and confidentiality using modern cryptography. But the IOT is not fitting for this strategy to be copied as is. The fundamental reason to it is that most of those billions of things are cheap, simple devices, which may cost a couple of bucks, and which may be installed and launched, not to be touched again. They are not designed to carry on their back a fanciful computer processor that can crunch the complicated number theoretic algorithms that underlie modern cryptography. What's more, these devices are powered by small batteries, which would be readily drained by a latched-on computer churning the prevailing algorithms.

So, what's the alternative -- to step back to pre-computer simple (very breakable) cryptography?

Not necessarily. We may exploit another technological miracle -- the means to store many gigabytes of bits in a cheap, tiny flash memory card. IOT devices cannot carry sophisticated computers, which drain their batteries too fast, but they can easily and cheaply be fitted with oodles of random bits.

Randomness and Cryptography. Cryptography feeds on randomness: it takes in the 'payload' -- the stuff that needs to be protected, mixes it with some random bits, and then issues the protected version of the payload. This can be written as follows: security is generated by using some measure of randomness and applying data "mixing" over the payload to be protected, and the random input. Now, historically, researchers opted to use as little randomness as possible, and build the required security by more elaborate data mixing. Since mixing is an energy hog, while randomness is passive affordable resource, it stands to reason that to meet this new challenge we might look for easy data mixing compensated with large amounts of affordable, easy to use, randomness.

This new strategy towards IOT security will keep this sensitive network secure against even very vicious attacks.

There is a whole suite of ciphers that are a result of the new strategy. The reader is pointed to the reference citings below [Samid 2002, 2004, 2015A, 2015B, 2016C]. In this piece we focus on a simple very common task -- verifying a prover.

Verifying an IOT Client

IOT sensors and controllers serve clients who consume their readings, and who send them behavioral instructions. The IP protocol gives access to the rest of the network and it tempts all sorts of abusers either to read readings that they should not, or to issue commands that would be harmful. It is therefore necessary for the IOT device to verify that it deals with its client, and no other.

There are numerous prover-verifier protocols to choose from but athey are computing-heavy, and battery hogs. We are seeking a cheap "data mixer" combined with cheap storage technology to generate the necessary security.

The sections ahead describe a proposed solution.

Security Based On Large Secret Quantities Of Randomness

Our aim is to generate security by exploiting modern memory technology, while relying on minimum computational power. We will do it by relying on much larger quantities of randomness than has been the case so far, and by limiting ourselves to basic computational primitives that are easily implemented in hardware.

Modern ciphers rely on a few hundreds or a few thousands of random bits. We shall extend this ten, or hundred fold and beyond. We have the technology to attach to an IOT device more than 100 gigabytes of randomness. On the computation side we will use simple bit-wise primitives like *'compare', 'count'*, and *'flip'.*

A typical IOT device will easily be engineered to add another important element for its operation: ad-hoc non-algorithmic randomness. Say, a temperature sensor, reading ambient temperature at intervals Δt. Random environmental effects will move the reading up and down. A simple computing device will generate a "1" each time the present reading is higher than the former reading, and generate a "0" otherwise. This raw bit-string will then be interpreted as follows: a combination of "01" will be regarded as a "0"; a combination of "10" will be regarded as "1", combinations of "00" and "11" will be disregarded. This will generate a uniform randomized string. This string is not pre-shared of course, but also immunized from theft because it was generated just when it was needed, not before (ad-hoc). It is easy to see that even if the environment cools, or heats up this method will work. If the environment heats up then there will be more "1" than "0" in the raw string, or say Pr(1) > Pr(0): the probability for "1" to show up next is higher than the probability of a "0" to show up next. However the probability of a pair of zero and one is the same regardless of the order:

$$Pr(\text{"01"}) = Pr(\text{"0"})*Pr(\text{"1"}) = Pr(\text{"1"})*Pr(\text{"0"}) = Pr(\text{"10"})$$

As to philosophy of operation we now build upon a modern concept of probability based security. Common protocols, like 'zero knowledge' types, are based on allowing the parties to replace the old fashioned message certainty with at-will probability, which in turn creates a corresponding at-will probability for adversarial advantage. We elaborate:

Cryptography is key based discrimination between those in possession of that key and all the rest. A lucky guess can produce any key and wipe out this discrimination. Security is based on the known, calculable and well managed low probability for that to happen. The unadvertised vulnerability of modern cryptography is that the apparent probability for spotting the key may be much higher than the formal one: 2^{-n} for an n bits string. The complex mathematics of modern ciphers may be compromised with a clever shortcut, as has happened historically time and again. By avoiding complex algorithms one removes this vulnerability.

We also propose to exploit probability at the positive end and make greater use of it at the negative end. Nominally Alice sends Bob a message which Bob interprets correctly using his key. There is no uncertainty associated with Bob's interpretation. What if, we induce a controlled measure of uncertainty into Bob's reading of the message? Suppose we can control this uncertainty to be as low as we wish (but still greater than zero). And further suppose that in the highly unlikely case where the residual uncertainty will prevent Bob from a proper interpretation of the message, then he will so realize, and ask Alice to try again? Under these circumstances it will not be too costly for us to replace the former certainty with such a tiny uncertainty, and will do it if the pay off justifies it. It does -- the tiny uncertainty described above (at Alice's end -- the positive end) will loom into a prohibitive uncertainty facing Eve who tries to win Bob's false verification. And that's the trade that we propose.

Come to think about it, modern zero knowledge dialogue use the same philosophy -- a small uncertainty at the positive end buys a lot of defensive uncertainty at the negative end.

Randomness Delivers

The brute force approach to solving the Traveling Salesman problem for finding the shortest trail to visit n destinations when all n^2 distances are specified is O(n!) -- super exponential. Yet, prospectively, it can be solved with $O(n^2)$ because the n^2 distances between the traveled destinations do determine the answer, which means that one must take into account the specter where a smart enough mind finds this shortcut and solves the traveling salesman problem at $O(n^2)$. The traveling salesman is regarded as an anchor problem for many intractability based security statements, and all these statements face the same vulnerability offered by yet unpublished mathematical insight.

If, on the other hand, one of the n! possible sequences of order of the n destinations is randomly selected, then there is no fear of some fantastic wisdom that would be able to spot this random selection on average in less than n!/2 trials. In short: randomness delivers guaranteed security, and is immunized against superior intelligence.

In this particular randomness bit-flipping protocol security is based on hard core combinatorics. The probability for a positive error (clearing a false prover), and the probability for a negative error (rejecting a bona fide prover) are both firmly established, The users know what is the risk that they are takings.

The Randomness Approach to the Verifier-Prover Challenge

The simple way for a prover to prove possession of a shared secret Sec=S is to forward S to the verifier. That would insure (with nominal certainty) that the prover holds S. Alas, the verifier and prover communicate over insecure lines so Eve can capture S, and become indistinguishable from the prover. Casting this situation in terms of the present risk, $\rho_{present}=0$, versus the future risk, $\rho_{future}=1.00$ where a risk $\rho=1.00$ is regarded as the upper bound. This is clearly a shortsighted strategy. The standard solution to this deficiency is to use a different input, d, to compute a different derived shared secret, S_d, for each session. It is done in the following way: Let OWF be some one-way function which takes the secret Sec=S and an arbitrary d (not previously used) to generate an output q=OWF(S,d). The verifier selects d, notifies the prover, who computes q and conveys it to the verifier. The verifier will be readily persuaded that q was computed from S, accepting a risk of $\rho=1/|q|$ where |q| is the size of the set of all possible q values (technically true if d is randomly selected from its space). OWF and |q| may be selected to keep this risk lower than any desired level. Since each verification session is carried out with a previously unused d it so happens that Eve cannot use a former q value to cheat her way in. Ostensibly her chances to guess q right are the same each successive round: 1/|q|. Alas, this analysis ignores the possibility that the selected OWF will be cracked -- namely, will become a two-way function. In that case Eve will reverse compute S from the former q, and again become indistinguishable from the prover.

We may contrast the above strategy with the one where the prover would resort to a random value, r, and use it to compute q=RND(S,r), via a random-data processing algorithm RND, then convey q (without r) to the verifier. The verifier, aware of RND and S, but not of r, will have to conclude whether the sender of q is in possession of S or not. Two kinds of mistakes are possible: verifying an imposter, and rejecting a bona fide prover. This amounts to the risk of the present $\rho_{present}$.

Having exercised this protocol t times, Eve, the eavesdropper, would be in possession of t q values: q_1, q_2,....q_t. This possession will increase the chance for Eve to successfully send the verifier q_{n+1}. This information leakage will imply a growing future risk ρ_{future}.

Given any RND procedure the Verifier will be able to use solid combinatorics to credibly assess the two risks: $\rho_{present}$, and ρ_{future}, and balance between them. Generally the higher $\rho_{present}$, the lower ρ_{future}, and vice versa. It is a matter of a selection of a good RND procedure to improve upon these risks and properly balance between them.

This randomness based procedure is not vulnerable to some unpublished mathematical insight because algorithmic complexity is not relied upon in assessing security.

Whatever the present risk ($\rho_{present}$), the randomness based procedure may be replayed as many times as necessary, and thereby reduce the risk at will. By replaying the procedure n times the risk becomes $\rho^n_{present}$. This "trick" does not work for solutions based on algorithmic complexity. If the algorithm is compromised then it would yield no matter how many times it is being used.

RND procedures are also computationally simple, while one way functions tend to be very burdensome from a computational standpoint, which gives a critical advantage to randomness based security when the verifier is a device in the Internet of Things, powered by a small battery or by a small solar panel. IOT devices equipped with powerful computers are also a ripe target for viral hacking, as recently argued [Ronen 2016]. Simple ad-hoc computers will neuter this risk.

Conditions for an IOT-friendly Effective Prover-Verifier Protocol

Let Alice and Bob share a secret Sec=S for the purpose of identifying one to the other. S is a bit string comprised of s bits. Alice and Bob may be human entities or represent 'devices' operating within the Internet of Things (IoT). Bob needs to find a way to convince Alice that he is in possession of S (and hence is Bob), but do so in a way that Eve, the eavesdropper will not be able to exploit this event to successfully impersonate Bob.

Opting for a probability based strategy, Bob will send Alice a "proof of possession of S", Prf = P, where P is a bit string comprised of p bits ($P = \{0,1\}^p$). This protocol will have to comply with the following terms:

1. **Persuasiveness:** Alice, the verifier, receiving P will reach the conclusion that prover Bob's version of Sec=S_p = S:

$$\text{(1)..... } \Pr[\, S \neq S_p \mid Prf = P] \rightarrow 0 \text{ for } s,p \rightarrow \infty$$

2. **Leakage:** Eavesdropper Eve, reading Prf=P will face a sufficiently small probability to establish her version of Sec= S_e such that S_e = S:

$$\text{(2)....... } \Pr[\, S = S_e \mid Prf = P] \,] \rightarrow 0 \text{ for } s,p \rightarrow \infty$$

Persuasiveness and leakage are the common and necessary probabilities for a prover-verifier dialogue. Albeit, we introduce a third term: abundance of proofs:

$$\textbf{(3)...... } \Pr[Prf=P \mid Sec=S] \rightarrow 0 \text{ for } s,p \rightarrow \infty$$

Namely, there is a large number of proofs Prf = P_1, P_2,... that will each persuade the verifier that the prover is in possession of S.

This feature of "abundance of proofs" allows the protocol to use a durable secret S, and also to detect hacking attempts. Suppose for a given Sec=S there would have been only one proof Prf=P. In that case Eve would read P as it sails through the veins of the Internet, and replay it to Alice, persuading her that she is Bob without ever knowing the shared secret S. And because of that Alice and Bob would have to use Sec=S to generate a derived per session secret S, S', S"... so that learning the identity of P in proving possession of one (or several) session keys would not be useful for Eve to arrive at the correct value of Sec = S = S_e. Since the derivation formula S → S', S",... will have to be exposed, then Alice and Bob will have to rely on this formula to be a one-way type in order to benefit from this feature. "Onewayness" relies on algorithmic complexity though, and introducing it will stain the purity of the solution so far which is immunized towards further mathematical insight.

On the other hand, the abundance of proofs may be used by Bob, the prover, through randomly selecting one valid instance of the Prf set: Prf = P_i i=1,2,.... each time he needs to prove his identity of Alice (through proving to her he holds the secret Sec = S = S_p). Alice will keep a log of all the proofs P_1, P_2,... that were used before, and if any of these proofs is replayed ("as is" or with slight modification) then Alice will first spot, it, and second will be on the alert that Eve who eavesdropped on the her previous communications with Bob, is seriously trying to hack into her.

We will now present a procedure that satisfies all these three conditions.

The Bit-Flip Protocol

We first describe the basic idea of the "Bit Flip" protocol, then we build on it.

Alice and Bob share a secret Sec=S comprised of s bits, where the value of s is part of the secret. At some later point in time Bob wishes to communicate with Alice, so Alice wishes to ascertain Bob's identity by giving Bob the opportunity to persuade her that he is in possession of S, without ever communicating S over the insecure lines they are operating at. To that end Alice picks an even number $p < s$ and sends that number to Bob. Bob, in turn, randomly cuts a p-bits long substring, S_p, from S: $S_p \subset S$. Then Bob -- again, randomly -- flips half the bits in S_p to generate the proving string P, which he sends to Alice in order to prove his possession of S.

Upon receipt of P Alice overlays the string with respect to S assuming that S_p starting bit was the first bit in S. She then checks if the p-bits long overlaid substring of S, S[1,p], which is stretching from bit 1 in S to bit p in S is the same as the string Bob sent her, P, apart from *exactly* p/2 bits which are of opposite identity. If indeed P and S[1,p] share p/2 bits and disagree on the other p/2 bits then Alice concludes that Bob is in possession of their shared secret Sec=S. If not then Alice compares P with S[2,p+1] – the p-bits long substring of S which starts at bit 2 on s and ends at bit p+1 in S. If the comparison is positive then Alice verifies Bob. If not Alice continues to check P against all the p-long substrings in S. If any such substrings evaluates as a positive comparison with P then Alice verifies Bob, otherwise she rejects him.

To build a nomenclature we define an operation *Rflip* as follows: Let X be an arbitrary bit string comprised of x bits. Operating on X with $Rflip_n$ for $n \leq x$ amounts to randomly flipping n bits in X to generate a string X_f also comprised of x bits:

$$(4)........X_f = Rflip_n\, X$$

One may note that R*flip*$_n$ X ≠ R*flip*$^n_1$ X because by applying R*flip* n times on X there is a chance that a previously flipped bit will be flipped back. With this nomenclature we can write that Alice will verify Bob if P satisfies the following condition:

$$(5)\ldots\ldots\ldots\ldots P = Rflip_{0.5p}\ S[i,i+p] \text{ for some } i \text{ from } 1 \text{ to } s\text{-}p.$$

Since flipping is symmetric, the following equation expresses the same as the former:

$$(6)\ldots\ldots\ldots\ldots S[i,p+i] = Rflip_{0.5p}\ P\ ] \text{ for some } i \text{ from } 1 \text{ to } s\text{-}p$$

Properties of the Bit-Flip Protocol

The salient feature of the Bit-Flip protocol is that it avoids any reliance on algorithmic complexity. The entire protocol is based on randomized processes. Which means that to the extent that the deployed randomness is 'pure' the chance for a mathematical shortcut is zero. Or say, the only threat for breaking the security of the BF protocol is the possibility (perhaps) of applying ultra fast computing machinery.

Furthermore, the actual security projected by the protocol is fully determined by the user upon selecting the values of |S|=s, and |P|=p, plus, of course, deploying quality randomness. As we shall see below the level of confidence to be claimed by Alice for correctly concluding that the party claiming to be Bob is indeed Bob (meaning is in possession of their shared secret Sec=S) is anchored on solid probability arguments. In other words, the BF protocol allows for an exact appraisal of the persuasiveness condition, as well as the exact appraisal of the leakage condition. As to the abundance condition it is clear by construction that Bob has a well calculated large number of possible proofs, P, to prove to Alice that he is in possessions of S.

In summary, the BF protocol satisfies the persuasiveness condition, the leakage condition and the abundance condition and thereby qualifies as an IOT-friendly prover-verifier protocol.

Combinatorics

Let us first check the simple case where s=p, namely, Bob, the prover, picks the full size of S (which we assume to be comprised of even number of bits) to generate the proving string P. Bob has $|Prf| = p!/(0.5p)!^2$ possible proofs such that each of these proofs $P_1, P_2, \ldots P_j$ for j=1 to j =|Prf| will be a solution to the equation:

$$(7)\ldots\ldots\ldots P_j = Rflip_{0.5p}S[1,s]$$

This expression is readily derived: the first bit to flip can be selected fro p (=s) options. The second from the remaining (p-1) bits, and the i-th bit to flip may be selected from (p-i) options, for i=0,1,....(0.5p-1) By so listing the various bit-flipped strings, we list every string (0.5p)! times, since they appear in all possible orders. So by dividing p(p-1).....(p-0.5p+1) by (0.5p)! we count the number of strings that would satisfy the equation above.

This is an abundance which is fully controlled by Alice and Bob by setting up the value of s (=p). Which means that if used correctly (namely randomly selecting p bits to flip) then the chance for Bob to use the exact proof twice may be made negligible, or as small as desired, by simply selecting the value of s. Say then, that if Alice keeps track of the successful proving strings P then when she spots a replay, she will be confident that it is fraudulent.

Eve who captured a proving string P will face a 50:50 chance for each bit in P to be what it is, or to be the opposite. And so she will enjoy a very meager leak, as computed ahead:

However, Eve could try to replay a modified P ($=P^m$) that would be sufficiently modified not to be rejected as a strict replay, but sufficiently similar to P to attack the protocol with a non-negligible chance to meet Alice acceptance criteria.

Should Eve flip two random bits in a previously qualified Prf=P, she will have a 25% chance to flip the pair such that the count of flipped bits will remain 0.5p, and hence Eve' modified string P^m might get her verified. However, Alice will find Eve's modified string to be too close to the P string she previously used to verify Bob. After all (p-2) bits are the same in the two strings. Alice will then deduce that Eve captured P and modified it to P^m. This will evoke her suspicion and she will either reject Eve outright, or use one of the methods (discussed ahead) to affirm her opinion (e.g. asking Eve to send another proving string). By flipping 4 bits, or 8 bits, Eve reduces her chance to be verified to 1/16 and 1/256 respectively, but still raise Alice's suspicion because so many other bits are the same in P and in P^m. Eve eventually might have in her possession some t previously verified strings, and based on this leaked knowledge, try to come up with a string that would be different from all the previous strings, but still have a non-negligible chance to be verified. Indeed so, but Alice has the same information at least. She knows the identity of the previously verified strings, so she too can appraise the chance that Eve's P^m string is a sophisticated replay of the old strings, and act accordingly. Both Eve and Alice in the worst case, are exposed to the same data, and much as Eve can appraise her chance to be falsely verified, so does Alice -- no surprises.

If Bob uses high quality ad-hoc randomness to generate his proving string P, then it would be 'far enough' from all the previously used t strings (the more so, for larger P).

Since every previously verified string P_i satisfies:

$$(8)\ldots\ldots\ldots P_i = Rflip_{0.5p}\ S$$

it is also true that:

$$(9)\ldots\ldots\ldots S = Rflip_{0.5p}\ P_i$$

This reduces the size of the set that includes S from 2^s to the set of all S values that satisfy the above equations for all i=1,2,...t

The size of the set F_i of S size strings that satisfy the Rflip equation for any P_i is:

$$(10)\ldots\ldots |F_i| = p! / (0.5p!)^2$$

Given a previously verified string P_i, Eve would be able to mark $|F_i|$ strings that include the secret Sec=S. (A-priori in the case where s=p, the secret S is known to be included in the full set comprised of 2^s members). After spotting the first verified string P_1, Eve would be able to limit the set that includes S to the F_1 set. The shrinking of the inclusive set of S represents the leakage.

Given t verified strings $P_1, P_2,....P_t$, the accumulated leakage amounts to further limiting the inclusive set for S according to the condition that S will have to be included in every one of the t F_i sets (i=1,2,...t):

$$(11)\ldots\ldots\ldots\ldots S \in (F_1 \cap F_2 \cap \ldots\ldots\ F_t)$$

This situation raises an interesting question. Given the set of t previously verified strings $P_1, P_2,....P_t$, Eve could apply the brute force approach to find good S candidates: she will randomly select an S string (out of the 2^s possibilities), and then check if that candidate, S_e, satisfies:

$$(12)\ldots\ldots\ldots S_e = RFlip_{0.5p}\ F_i \text{ for } i=1,2\ldots t$$

If any of these t equations is not satisfied, then the candidate should be dropped. By probing for all 2^s candidates Eve will generate the reduced set of S candidates from where she should randomly pick her choice. This is obviously a very laborious effort, especially for large enough s values. The question of interest is whether there is a mathematical shortcut to identify the reduced set of S candidates, based on the identity of the t verified strings. Be it what it may, for security analysis we shall assume that such mathematical insight is available and rate security accordingly.

The above attack strategy is theoretically appealing but may not be very practical if after the enormous work to identify the reduced S set, that set is still too large for Eve to have a non-negligible chance to select the right S (and hence use a successful proving string P). The 'flip a few' bits attack, discussed above seems a more productive strategy.

In summary, Alice is fully aware as to how much information has been leaked to a persistent eavesdropper who captured P_1, P_2,.....P_t and can accurately appraise the chance that Eve sent over P_e based solely on leaked information. It will then be up to Alice to set up a suspicion threshold, above which she will ask Bob to send another (and another if necessary) proving string, or ask Bob to flip back a specified number of bits (see discussion ahead).

Persuasiveness: The leakage formula above implies that if the leakage so far is small enough, then the chance that Alice will regard Eve as Bob is small enough, which in turn implies that if P $\epsilon\ Prf$ then the prover is Bob (or at least is in possession of the shared secret Sec=S).

In other words, Alice and Bob, using the Bit-Flip protocol, may select a secret Sec=S of size s bits large enough to insure a bound risk of compromise over an arbitrary number of captured previous proving strings.

All that was over the simple (and most risky) case where p=s. The leakage becomes increasingly smaller for p < s. Albeit, the persuasiveness is also smaller.

In the general case where s > p Bob can choose (s-p) subsets to apply R*flip* over. This will imply that the Prf set is larger, and thereby the blind chance to randomly select a proving string P such that P $\in$ Prf is larger. However it can still be maintained below a desired level δ.

We concluded that for s=p the size of Prf is given by:

$$(13)\ldots\ldots\ |Prf|_{s=p} = p!/(0.5p!)^2$$

For s>p there are (s-p) situations similar to s=p, and hence:

$$(14)\ldots\ldots\ |Prf|_{s>p} \leq (s-p)(|Prf|_{s=p}) = (s-p)p!/(0.5p!)^2$$

The probability for a per chance proving string to pass as bona fide is given by:

$$(15)\ldots\ldots\ Pr[Prf=P|\ S \neq Sec] = |Prf|_{s>p} / 2^p = 2^{-p}(s-p)p!/(p!)^2$$

And since both s and p are selected by Alice and Bob, so is the risk that Alice faces to be falsely persuaded.

For example for s= p=40: The number of bona fide proving strings |Prf|=137,846,528,820, and the chance for Eve to select a $P_e \in$ |Prf| is:

$$(16)\ldots\ldots\ \rho_{present} = Pr[Prf = P_e \mid p=s=40] = 137846528820 / 2^{40} = 0.125$$

This is clearly too high for comfort, and remedy is called for. It may be in the simplest form of replay. If the verifier asks the prover to repeat the process, say 5 times then the probability for Eve to be accepted as Bob will shrink to $3.1*10^{-5}$

The leakage after one round will be quite limited. Eve, realizing that P was used to verify Bob, will then be able to limit the space from which to choose, from 2^s to $p!/(0.5p!)^2$, so the added risk for the verifier to be cheated is:

$$(17)\ldots\ldots\ldots p_{future}(1) = 1/p!/(0.5p!)^2) - 1/2^s = 1/137846528820 - 1/1099511627776 = 10^{-11}$$

This negligible risk will rise dramatically after t > 1 rounds, since the number of proving strings to choose from will be limited to those strings that would be admissible versus all t proving strings.

We shall now examine two add-on elements to this basic procedure: (1) s > p, and (2) The Re-Flip Strategy.

The s>p Strategy

When analyzing the case where the shared secret Sec=S is as large as the proving string P (|S|=s=|P|=p), we concluded that the accumulated list of verified strings P_1, P_2,... P_t effected a leakage that Eve could exploit to improve her chances to pass to Alice a bona fide string $P_e \in$ Prf. We concluded that by increasing the size of the proving string (equals the size of the secret), the chance for Eve to randomly pick a bona fide proving string was reduced, but at the same time the leakage increased too, threatening the future performance of the protocol.

This threat of increased leakage can be properly answered by the "s>p" strategy. Alice and Bob may share a secret S of size |S|=s bits larger than the prover string P of size |P|=p bits (s>p).

The "pure" way to accomplish this is to set |S| = n*|P|, where n=2,3,... This means that the shared secret will be a secret multiple of the size of the selected proving string.

Bob will then randomly choose one of the n p-size strings, apply the $RFlip_{0.5p}$ operator to it, and send the result over to Alice. Alice will check each one of the n strings to see if the string Bob sent qualifies as belonging to Prf for any one of the n options. If it does, then Alice verifies Bob.

A somewhat less "pure" way for accomplishing the same is to set $|S| = |P| + n$, where n=1,2,... Bob will then pick a subset of S ($S_p \subset S$), and apply $Flip_{05p}$ to it, to generate a proving string, P, for Alice to evaluate. Alice will check if the proving string P qualifies for any of the n subsets in S. If it does, then Alice verifies Bob. Otherwise Alice rejects him.

This simple twist will stop the leakage. As long as Eve does not know the size of the shared secret Sec=S, she cannot link the information from the t previously verified proving strings because for any two previously verified proving strings Eve would not know whether they are the result of Rflip application to the same base string or not. If Eve somehow finds out the size of the shared secret and the method in which it is being parceled out to base strings to apply RFlip over, then she can apply some useful combinatoric calculus. But even in this case, a modest over size s>p will build a very robust security, which like before, is very accurately appraised by Alice.

By allowing for every proving string, P, to qualify over any of the n options afforded by the "s>p strategy" Alice increases the risk for Eve to randomly pick a bona fide proving string $P_e \in Prf$. The probability for such pick will be an n-multiple of the s=p probability:

$$(18).......Pr[P_e \in Prf \mid |S|=n*|P|] = 1-(1-(p! / ((0.5p)!^2 * 2^p))^n$$

which should not pose any serious problem because Alice and Bob can select S and P such that this risk will be below any desired threshold.

In summary, the "s>p" strategy, stops the leakage of the "s=p" strategy, and does so at a very reasonable cost of proper bit size for the shared secret Sec=S and for the proving string P.

Note: the above discussion is limited to Bob flipping half of the bits in the flipped string. This ratio may also be changed. Bob can be asked by Alice to flip only a quarter, or only, say 50 bits in the flipped strings. This will affect the results, but will not fundamentally modify the equations.

The Re-Flip Strategy

Alice in essence tries to distinguish between a proving string P sent to her by Bob to prove his possession of their shared secret Sec=S, and between Eve who is using the history of the Alice-Bob relationship to successfully guess a qualifying proving string P. One way to so distinguish is to ask a follow up question that references the flipped bits in P. Bob would know which bits he flipped, but Eve will not. The question may be a simple re-flip: Alice asks Bob to flip back some f' bits in P -- that is to undo the original flipping over a random choice of $f' < 0.5p$ bits. Of course if $f'=0.5p$ then Bob will flip back all the bits he originally flipped and thereby expose S. So f' must be quite small, yet large enough to suppress the chance for Eve to successfully respond to this challenge.

There is an infinite number of questions that Alice can ask with relevance to the flipped bits. Some may be quite sophisticated and allow for only minimal information leakage. But again, the important point is that for any such question Alice and Bob can credibly appraise both the present risk ($\rho_{present}$), and the future risk (ρ_{future}) of their connection.

The Re-Flip strategy comes with a cost. When Bob submits to Alice the identity of the requested f' flipped bits, he also signals to Eve what the identity of these f' bits is, so from now on Eve is in doubt only with respect to s-f' bits in S. If this scheme is used some k times then the effective size of S becomes s-f'k. This cost too can be mitigated by a proper choice for s and f'. If Bob successfully identifies f' flipped bits then the chance that he guessed his answer is $1/2^{f'}$ which should be multiplied by the previous risk for falsely verifying Bob: $\rho_{after} = \rho_{before} /2^{f'}$ So for s=100,000, a value of f'=10 will reduce the risk for an error by a factor of 1024, and if applied, say 1000 times, then, at most the effective size of S will drop to 90,000 bits.

A more sophisticated variation on the re-flip strategy is to ask several questions with known probability of guessing, but such that they do not identify the identity of any bit. For example: (1) what is the distance in bits between the two furthest apart flipped bits, (2) how many pairs of flipped bits are x bits apart?, or (3) what is the sum of the bit position count of all the flipped bits.

Illustration: Let s=p=8, and let S=10110111. There are $70 = 8!/(4!)^2$ possible proving strings for Bob to send Alice (|Prf|=70) which represents a fraction of 27% out of the 2^8=256 possible strings of size eight bits. This is too risky, so Alice resorts to the Re-Flip strategy. In its basic form Alice asks Bob to flip back 2 bits. While Bob will do so accurately, Eve would have a ¼ chance to guess correctly, and this would reduce the risk for Alice to falsely verify Eve to 0.27/4= 0.067, but then reduce the effective size of the shared secret to 6 bits. Suppose that the proving string that Bob sent to Alice was: P= 10000010, namely Bob flipped bits: 3,4,6,8. If Alice asks for the sum of the positions of the flipped bits, Bob will answer: 3+4+6+8 = 21.

Numbers

In this section we present the Bit-Flip protocol with numbers. We first refer to the case where s=p: |S|=|P|. The table below lists the size of Prf -- the set of all the bona fide strings, namely the strings that satisfy the equation: P = $\text{Rflip}_{0.5p}$ S, as well as the risk ($\rho_{present}$) for Eve to randomly pick a bona fide proving string, on a single try, on five tries and ten.

		ρ-present		
s=p	**|Prf|**	**one round**	**five founds**	**10 rounds**
20	184756	0.18	1.69E-04	2.90E-08
50	1.26E+14	0.11	1.78E-05	3.18E-10
100	1.01E+29	0.08	3.19E-06	1.01E-11
250	9.12E+73	0.05	3.25E-07	1.06E-13
1000	2.70E+299	0.02	1.02E-08	1.04E-16

It is clear that for |P|=1000 bits, for example, the shared secret S may be 10^{12} times the size of the proving string, P, and the risk for a false verification will be in the range of 1/10000, on a protocol of Alice asking Bob to pass the test 10 times.

Implementing the Flip-Bit Protocol

Alice and Bob may conclude that modest values of secret size (|S|=s), and proving string size (|P|=p) will deliver accepted level of security as indicated by strict combinatorics calculation. They might decide on selecting of 'secret reservoir' (S_r) from where to chop off operational secrets of size |S|=s. The actual secret Sec=S may be pre-set for use on a fixed schedule, or perhaps be event driven. The existence of a large 'secret reservoir' offers Alice and Bob a great measure of operational flexibility. They can mutually decide to change (increase or decrease) the size of the verification secret, S, they can decide on changing the relationship between s and p (the size of the secret versus the size of the proving string), and of course, they can decide to use a new secret, at will.

Alice and Bob will be able to distinguish between a 'dumb attack', a 'learned attack' and a 'smart attack', and adjust their security accordingly. A dumb attack happens when Eve tries her luck with a random pick -- against which the odds are well established. A 'learned attack' happens when Eve tries to replay a previously successful proving string, P. It indicates to Alice and Bob that Eve is actively tracking them. A 'smart attack' happens when Eve uses limited and well thought out modifications of previously played proving strings to maximize her odds to be falsely verified. This is the most serious challenge to the system, but credible combinatorics will fend it off. If a proving string appears 'too close' to a previously used string, then Alice may request another one. Awareness of such attacks may be very useful for (1) cyber intelligence purposes, and (ii) for optimizing counter measures, like: it's time to switch to the next secret segment from the secret reservoir.

The security gained through randomness herein, can always be augmented through algorithmic complexity, for good measure. This option will be discussed ahead. Also, the ad-hoc randomness (r) used by Bob to generate the proving string P may then be used by Alice and Bob as per-session shared secret, see ahead.

The Bit-Flip protocol also requires ad-hoc non pre-shared randomness. This can be implemented in non-algorithmic ways using white noise apparatus.

Algorithmic Complexity Add-On

The randomness based security strategy described herein may be augmented at will with conventional algorithmic-complexity security. As indicated before, the secret, Sec=S, together with a per-session different number, d, serve as an input to a one-way function OWF to compute an outcome q, which is what Bob needs to prove to Alice he is in possession of. To the extent that OWF is compromised this strategy fails. However it is applied on top of the randomness strategy, that is the randomness strategy is applied over q, then algorithmic complexity serves as add-on security.

In choosing a robust OWF for IOT devices, the original constraint of light computation still applies. Most common OWF are number-theoretic and hard computing. A randomness based alternative is offered below:

One-Way Transposition

Aiming for a minimal computational solution for a robust one-way function, one might focus on the primitive of transposition, as follows: Let S be a bit string of size s. Let r be a positive integer regarded as the 'repeat counter'. Let us generate a permutation of S ($=S_t$) by applying the following procedure:

Consider a bit counting order over S such that when the count reaches either end of S it continues in the same direction but starting at the opposite end. Starting from the leftmost bit in S, count r bits left-to-right. The bit where the counter stopped will be pulled out of S, and placed as the rightmost bit of a new string, S_t. We keep referring to the former S string as S although it is now of size (s-1) bits $S=S[|S|=s-1]$. If the removed bit is '0' then keep counting r more bits, in the same direction. If the removed bit is "1" then switch direction: instead of right to left, keep counting left to right, and vice versa. Each bit that stops the counter is removed in turn from S and placed as the leftmost bit in S_t. The counter is eventually stopped s times, and by then S is empty $S=S[|S|=0]$ and $S_t = S_t[|S_t|=s]$ is bona fide permutation of S. Without the switch of direction of counting, given the value of the repeat counter r, it is easy to revise $S_t \rightarrow S$. But owing to the switching rule, it appears that brute force is the fastest way to reverse the permutation. And since the number of permutation is s!, it appears that reversing this "one-way transposition" routine is O(n!). Albeit, like other OWF, the risk of some hidden mathematical insight must be accounted for, and that is why OWF is recommended as a boost to randomized protection, not as a replacement thereto. See [Samid 2015B] for how to expand the above description to a complete transposition algorithm.

The table below summarizes the security enhancement options available for the Bit-Flip user:

Bit-Flip Strategy Options:

IOT devices span a large canvass of situations where cost, risk, network, exposure etc. do vary. The effort to insure security must fit into the economic picture. What we have shown, and what is summarized below is that the BF protocol may be implemented using a variety of security features. The basic s=p mode may be augmented simply by increasing the size of the shared secret Sec=S, and the size of the proving string Prf=P. It can be augmented by shifting to the "s > p" mode, even on a modest basis, the effect is very strong. The protocol might invoke the 'flip back' option – simple, powerful, and of course one might add today's practice of algorithmic-complexity in the form of a one way function. And whatever the configuration of the above strategies, by repeating the BF dialogue n times the risk is hacked down by the power of n.

Basic	Security Add-Ons				
s=p	larger S,P	Repeat	s>p	Flip-Back	OWF
x					
x	Any Combination of Security Add-Ons				
x					
x					
x	x	x	x	x	x

Per-Session Shared Randomness

The verified proving string, P indirectly communicated to Alice a random element, R. This element may be used for this session communication between Alice and Bob. It can be done directly, or as a part in a more involved protocol. The proving string P when contrasted with the pree-flipped string may define a formation bit string where each flipped bit will be marked one, and each unflipped zero. This is not a non-leakage secret, but still high entropy secret, and it may be used to XOR plaintext on top of whatever cryptography is applied to it. This strategy involves the risk that if the per-session secret is compromised somehow, then it would lead to losing the pre-flipped secret.

For example, let S=100010, and let Bob flipped bit 2,4,6, counting from right to left, resulting in P = 001000. The shared secret per session will be: 101010.

Randomness Management

Considering an array of IOT devices, it is common to manage them through a hierarchy. The hierarchy will have parent nodes and child-less nodes. The child-less nodes are the ones on the front line, and most vulnerable to a physical assault. Simple devices will not have too much protection against a hands on attacher, and one must assume that the protective hardware was compromised, exposing the device randomness. More critical devices might be designed with any of several options for erasure of the secret randomness upon any assault on its physical integrity. As to Differential Power Analysis (DPA) the Bit-Flip cryptography is much less vulnerable because it does not use the modular arithmetic that exposes itself through current variations. Yet, a Bit-Flip designer must account for the possibility of a device surrendering its full measure of randomness. This will void the communication ring shared by all the devices that work on the same secret randomness. It is therefore prudent to map the randomness to the functional hierarchy of the devices, rather than have one key (randomness) shared by all. We then envision every parent node to have three distinct Bit-Flip keys (randomness): a "parent key" with which to communicate with its parent device, a "sibling key" with which to communicate with its sibling devices, and a "child key" with which to communicate with its children nodes. A child-less node, will have the same except the "child key".

Summary Note

The Bit-Flip Protocol offers a practical effective tool for the prover-verifier challenge, especially attractive for Internet of Things devices. It lends itself to energy efficient fast hardware implementation because the algorithm is based on *bit-wise* primitives: *'compare', 'flip'*, and '*count*'. It gives its user the power to determine and credibly gauge the level of security involved (level of risk). The Bit Flip protocol removes the persistent shadow of compromising mathematical shortcuts. The specific Bit-Flip solution proposed here is a first attempt. This field is ready to be investigated for more efficient algorithms operating on the same principle of using randomness to create a gauged, small, well controlled verification uncertainty in order to achieve an extended and overwhelming uncertainty (confusion) for any attacker of the system.

The feature of Bit-Flip of being immunized against compromising mathematical shortcut should render it attractive also for most nominal prover-verifier applications.

Reference

Aron 2016 "A Quantum of Privacy" j. Aron New Scientist Volume 231, Issue 3088, 27 August 2016, Pages 16–17

Chaitin 1987: “Algorithmic Information Theory” Chaitin G. J. Cambridge University Press.

Hirschfeld 2007: "Algorithmic Randomness and Complexity" School of Mathematics and Computing Sciences, Downey, R, Hirschfeld, D. Victoria Univ. Wellington, New Zealand. http://www-2.dc.uba.ar/materias/azar/bibliografia/Downey2010AlgorithmicRandomness.pdf

Hughes 2016: "STRENGTHENING THE SECURITY FOUNDATION OF CRYPTOGRAPHY WITH WHITEWOOD'S QUANTUM-POWERED ENTROPY ENGINE" Richard Hughes, Jane Nordhold http://www.whitewoodencryption.com/wp-content/uploads/2016/02/Strengthening_the_Security_Foundation.pdf

Kamel 2016: "Towards Securing Low-Power Digital Circuit with Ultra-Low-Voltage Vdd Randomizers" ICTEAM/ELEN, Université catholique de Louvain, Belgium. http://perso.uclouvain.be/fstandae/PUBLIS/176.pdf

Niels 2008: "Computability and randomness" Niels A. The University of Auckland, Clarendon, Oxford, UK

Perlroth 2013: Perlroth Nicole, et el "N.S.A. Able to Foil Basic Safeguards of Privacy on Web" The New York Times, Sept 5, 2013 http://www.nytimes.com/2013/09/06/us/nsa-foils-much-internet-encryption.html?_r=0

Ronen 2016 "IoT Goes Nuclear: Creating a ZigBee Chain Reaction" Eyal Ronen( )∗, Colin O'Flynn†, Adi Shamir∗ and Achi-Or Weingarten∗ *PRELIMINARY DRAFT, VERSION 0.93 ∗Weizmann Institute of Science, Rehovot, Israel*

Samid 2001A: "Re-dividing Complexity between Algorithms and Keys" G. Samid Progress in Cryptology — INDOCRYPT 2001 Volume 2247 of the series Lecture Notes in Computer Science pp 330-338

Samid 2001B: "Anonymity Management: A Blue Print For Newfound Privacy" The Second International Workshop on Information Security Applications (WISA 2001), Seoul, Korea, September 13-14, 2001 (Best Paper Award).

Samid 2001C: "Encryption Sticks (Randomats)" G. Samid ICICS 2001 Third International Conference on Information and Communications Security Xian, China 13-16 November, 2001

Samid 2002: " At-Will Intractability Up to Plaintext Equivocation Achieved via a Cryptographic Key Made As Small, or As Large As Desired - Without Computational Penalty " G. Samid, 2002 International Workshop on CRYPTOLOGY AND NETWORK SECURITY San Francisco, California, USA September 26 -- 28, 2002

Samid 2003A: "Non-Zero Entropy Ciphertexts (Stochastic Decryption): On The Possibility of One-Time-Pad Class Security With Shorter Keys" G. Samid 2003 International Workshop on CRYPTOLOGY AND NETWORK SECURITY (CANS03) Miami, Florida, USA September 24 - - 26, 2003

Samid 2003B: "Intractability Erosion: The Everpresent Threat for Secure Communication" The 7th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI 2003), July 2003.

Samid 2004: "Denial Cryptography based on Graph Theory", US Patent #6,823,068

Samid 2009: "The Unending Cyber War" DGS Vitco ISBN 0-9635220-4-3 https://www.amazon.com/Unending-Cyberwar-Gideon-Samid/dp/0963522043

Samid 2013: "Probability Durable Entropic Advantage" G. Samid US Patent Application 13/954,741

Samid 2015A: "Equivoe-T: Transposition Equivocation Cryptography" G. Samid 27 May 2015 International Association of Cryptology Research, ePrint Archive https://eprint.iacr.org/2015/510

Samid 2015B: "The Ultimate Transposition Cipher (UTC)" G. Samid 23 Oct 2015 International Association of Cryptology Research, ePrint Archive https://eprint.iacr.org/2015/1033

Samid 2016A: "Shannon's Proof of Vernam Unbreakability" G. Samid https://www.youtube.com/watch?v=cVsLW1WddVI

Samid 2016C: "Cryptography of Things: Cryptography Designed for Low Power, Low Maintenance Nodes in the Internet of Things" G. Samid WorldComp-16 July 25-28 Las Vegas, Nevada http://worldcomp.ucmss.com/cr/main/papersNew/LFSCSREApapers/ICM3312.pdf

Samid 2016D: "Celebrating Randomness" G. Samid Digital Transactions Nov 2016, Security Notes

Samid 2016E: "Cryptography of Things (CoT): Enabling Money of Things (MoT), kindling the Internet of Things" G. Samid The 17th International Conference on Internet Computing and Internet of Things, Las Vegas July 2016 https://www.dropbox.com/s/7dc0bgiwlnm7mgb/CoTMoT_Vegas2016_kulam_Samid.pdf?dl=0

Samid 2016F "Randomness Rising" http://wesecure.net/RandomnessRising_H6n08.pdf

Samid, 2016G "Cryptography – A New Era?" https://medium.com/@bitmintnews/cryptography-the-end-of-an-era-eceb6b12d3a9#.qn810eadn

Schneier 1997: "WHY CRYPTOGRAPHY IS HARDER THAN IT LOOKS" Counterpane Systems http://www.firstnetsecurity.com/library/counterpane/whycrypto.pdf

Shamir 1981: "On the Generation of Cryptographically Strong Psuedo-Random Sequences" Lecture Notes in Computer Science ; 8th International Colloquium of Automata, Springer-Verlag

Shannon 1949: "Communication Theory of Secrecy Systems" Claude Shannon http://netlab.cs.ucla.edu/wiki/files/shannon1949.pdf

Smart 2016: "Cryptography Made Simple" Nigel Smart, Springer.

Vernam 1918: Gilbert S. Vernam, US Patent 1310719, 13 September 1918.

Williams 2002: “Introduction to Cryptography” Stallings Williams, http://williamstallings.com/Extras/Security-Notes/lectures/classical.html

Zhao 2011 Zhao G. et al "A novel mutual authentication scheme for Internet of Things" Modelling, Identification and Control (ICMIC), Proceedings of 2011 International Conference.