Draft.

PROTECTING COMMUNICATIONS AGAINST FORGERY

DANIEL J. BERNSTEIN

Abstract. This paper is an introduction to cryptography. It covers secret-key message authentication codes, unpredictable random functions, public-key secret-sharing systems, and public-key signature systems.

1. Introduction

Cryptography protects communications against espionage: an eavesdropper who intercepts a message will be unable to decipher it. This is useful for many types of information: credit-card transactions, medical records, love letters.

There is another side to cryptography. Cryptography protects communications against sabotage: a forger who fabricates or modifies a message will be unable to fool the receiver. This is useful for *all* types of information. If the receiver does not care about the authenticity of a message, why is he listening to the message in the first place?

This paper explains how cryptography prevents forgery. Section 2 explains how to protect n messages if the sender and receiver share $128(n+1)$ secret bits. Section 3 explains how the sender and receiver can generate many shared secret bits from a short shared secret. Section 4 explains how the sender and receiver can generate a short shared secret from a public conversation. Section 5 explains how the sender can protect a message sent to many receivers, without sharing any secrets.

2. Unbreakable secret-key authenticators

Here is a protocol for transmitting a message when the sender and receiver both know certain secrets:

$$\begin{tikzcd}
 & \text{Secrets } p,k \arrow[d] & & \text{Secrets } p,k \arrow[d] \\
\text{Message } m \arrow[r] & \begin{array}{c}\text{Authenticated}\\ \text{message } m,a\end{array} \arrow[r] & \begin{array}{c}\text{Possibly forged}\\ \text{message } m',a'\end{array} \arrow[r] & \text{Verification}
\end{tikzcd}$$

The message is a polynomial $m \in F[x]$ with $m(0) = 0$ and $\deg m \le 1000000$. Here F is the field $(\mathbf{Z}/2)[y]/(y^{128} + y^9 + y^7 + y^2 + 1)$ of size 2^{128}. The secrets are two independent uniform random elements p and k of F.

The sender transmits (m, a) where $a = m(p) + k$. The forger replaces (m, a) with (m', a'); if the forger is inactive then $(m', a') = (m, a)$. The receiver discards (m', a') unless $a' = m'(p) + k$.

The extra information a is called an **authenticator**.

Date: 20010731.
1991 *Mathematics Subject Classification.* Primary 94A62.

Security. I claim that the forger has chance smaller than 2^{-108} of fooling the receiver, i.e., of finding (m', a') with $m' \neq m$ and $a' = m'(p) + k$. The proof is easy. Fix (m, a) and (m', a'), and count pairs (p, k):

- There are exactly 2^{128} pairs (p, k) satisfying $a = m(p) + k$. Indeed, there is exactly one possible k for each possible p.
- Fewer than 2^{20} of these pairs also satisfy $a' = m'(p) + k$, if m' is different from m. Indeed, any qualifying p would have to be a root of the nonzero polynomial $m - m' - a + a'$; this polynomial has degree at most 1000000, so it has at most $1000000 < 2^{20}$ roots.

Thus the conditional probability that $a' = m'(p) + k$, given that $a = m(p) + k$, is smaller than $2^{20}/2^{128} = 2^{-108}$.

In practice, the receiver will continue listening for messages after discarding a forgery, so the forger can try again and again. By flooding the receiver with a billion messages per second for a billion years, a persistent, wealthy, long-lived forger would be able to try nearly 2^{85} forgeries. His chance of success—his chance of producing at least one (m', a') with $a' = m'(p) + k$ and with m' not transmitted by the sender—is below 2^{-23}.

Handling many messages. One can use a single p with many k's to protect a series of messages:

$$\begin{tikzcd}
 & \begin{array}{c}\text{Secrets}\\ p, k_1, k_2, \ldots\end{array} \arrow[d] & & \begin{array}{c}\text{Secrets}\\ p, k_1, k_2, \ldots\end{array} \arrow[d] \\
\begin{array}{c}\text{Message } m,\\ \text{number } n\end{array} \arrow[r] & \begin{array}{c}\text{Authenticated}\\ \text{message } n, m, a\end{array} \arrow[r] & \begin{array}{c}\text{Possibly forged}\\ \text{message } n', m', a'\end{array} \arrow[r] & \text{Verification}
\end{tikzcd}$$

The sender and receiver share secrets $p, k_1, k_2, k_3, \ldots$. The sender transmits the nth message m as (n, m, a) where $a = m(p) + k_n$. The receiver discards (n', m', a') unless $a' = m'(p) + k_{n'}$.

In this context n is called a **nonce** and a is again called an **authenticator**. The random function $(n, m) \mapsto m(p) + k_n$ is called a **message authentication code** (MAC).

The forger's chance of success is at most $2^{-108}D$, where D is the number of forgery attempts. This is true even if the forger sees all the messages transmitted by the sender. It is true even if the forger can influence the choice of those messages, perhaps responding dynamically to previous authenticators. In fact, is is true even if the forger has complete control over each message, although a forger who starts with this much power obviously does not need to modify messages in transit!

History. Gilbert, MacWilliams, and Sloane in [16, section 9] introduced the first easy-to-compute unbreakable authenticator, using a long shared secret for a long message. Wegman and Carter in [31, section 3] proposed the form $h(m) + k_n$ for an authenticator and pointed out that a short secret could handle a long message.

There is now a huge literature on unbreakable MACs. For surveys see [22] and my paper [6]. For two state-of-the-art systems see [9] and [6].

3. Unpredictable random functions

Here is a protocol that is *conjectured* to protect a series of messages:

$$\begin{tikzcd}
\text{Secret } s \arrow[r] & p, k_1, k_2, \dots \arrow[d] & \text{Secret } s \arrow[r] & p, k_1, k_2, \dots \arrow[d] \\
\begin{array}{c}\text{Message } m,\\ \text{number } n\end{array} \arrow[r] & \begin{array}{c}\text{Authenticated}\\ \text{message } n, m, a\end{array} \arrow[r] & \begin{array}{c}\text{Possibly forged}\\ \text{message } n', m', a'\end{array} \arrow[r] & \text{Verification}
\end{tikzcd}$$

The sender and receiver share a secret uniform random 256-bit string s. The sender and receiver compute $p = \mathrm{MD5}(s, 0)$, $k_1 = \mathrm{MD5}(s, 1)$, $k_2 = \mathrm{MD5}(s, 2)$, etc. The sender transmits the nth message m as (n, m, a) where $a = m(p) + k_n$. The receiver discards (n', m', a') unless $a' = m'(p) + k_{n'}$.

MD5 is the function defined in [25], producing 128-bit output. The definition is much too complicated to be repeated here.

An attacker, given several authenticated messages, might try to solve for s. Presumably there is only one choice for s consistent with all the authenticators. However, the fastest *known* method of solving for s is to search through all 2^{256} possibilities. This is far beyond the computer power that will be available in the foreseeable future.

Is there a faster attack? Perhaps. We believe that this protocol is unbreakable, but we have no proof. On the other hand, this protocol has the advantage of using only 256 shared secret bits to handle any number of messages.

Unpredictability. Let u be a uniform random function from $\{0, 1, 2, \dots\}$ to F. Consider oracle algorithms A that print 0 or 1. What is the difference between

- the probability that A prints 1 using $n \mapsto \mathrm{MD5}(s, n)$ as an oracle and
- the probability that A prints 1 using u as an oracle?

It is conjectured that the difference is smaller than 2^{-40} for every A that finishes in at most 2^{80} steps. In short, $n \mapsto \mathrm{MD5}(s, n)$ is conjectured to be **unpredictable**.

If $n \mapsto \mathrm{MD5}(s, n)$ is, in fact, unpredictable, then this authentication protocol is unbreakable: a fast algorithm that makes D forgery attempts cannot succeed with probability larger than $2^{-105}D + 2^{-40}$.

History. Turing introduced the concept of unpredictability in [29]: "Suppose we could be sure of finding [laws of behaviour] if they existed. Then given a discrete-state machine it should certainly be possible to discover by observation sufficient about it to predict its future behaviour, and this within a reasonable time, say a thousand years. But this does not seem to be the case. I have set up on the Manchester computer a small programme using only 1000 units of storage, whereby the machine supplied with one sixteen figure number replies with another within two seconds. I would defy anyone to learn from these replies sufficient about the programme to be able to predict any replies to untried values."

The literature is full of very quickly computable short random functions that seem difficult to predict. Here **short** means that the random function is determined by a short uniform random string. Unfortunately, most of these random functions are "block ciphers" burdened by the unnecessary constraint of invertibility. See the books [27] and [19] for descriptions of many "block ciphers" and "random-access stream ciphers."

Blum, Blum, and Shub in [12] constructed a fast short random function with a small input, and proved that any fast algorithm to predict that function could be turned into a surprisingly fast algorithm to factor integers. Naor and Reingold in [21] constructed fast random functions with large inputs and with similar guarantees of unpredictability. These functions are never used in practice, because they are not nearly as fast as MD5; but they show that unpredictability is not a silly concept.

Unpredictability has an interesting application to complexity theory: one can use it to turn fast probabilistic algorithms into reasonably fast deterministic algorithms. This was pointed out by Yao in [33]. It is now widely believed that the complexity classes BPP and P are identical. See [17].

Beware that the name "unpredictable" has several aliases in the literature. See section 2 of my paper [4] for further discussion.

4. Public-key secret sharing

Here is a protocol for the sender and receiver to generate a 256-bit shared secret from a public conversation:

$$
\begin{tikzcd}
\begin{array}{c}\text{Sender's}\\ \text{secret } b\end{array} \arrow[d] \arrow[dd, bend right=60] & \begin{array}{c}\text{Receiver's}\\ \text{secret } c\end{array} \arrow[d] \arrow[dd, bend left=60] \\
\begin{array}{c}\text{Sender's}\\ \text{public key}\\ 4^b \bmod \ell\end{array} \arrow[dr] & \begin{array}{c}\text{Receiver's}\\ \text{public key}\\ 4^c \bmod \ell\end{array} \arrow[dl] \\
\begin{array}{c}\text{Shared secret } s =\\ \mathrm{SHA}(1, 16^{bc} \bmod \ell)\end{array} & \begin{array}{c}\text{Shared secret } s =\\ \mathrm{SHA}(1, 16^{bc} \bmod \ell)\end{array}
\end{tikzcd}
$$

The sender starts from a secret uniform random integer b with $0 \le b < 2^{256}$. The sender computes and announces a **public key** $4^b \bmod \ell$. Here ℓ is the prime number $\lfloor 2^{1534}\pi \rfloor + 444896$; note that $(\ell - 1)/2$ is also prime.

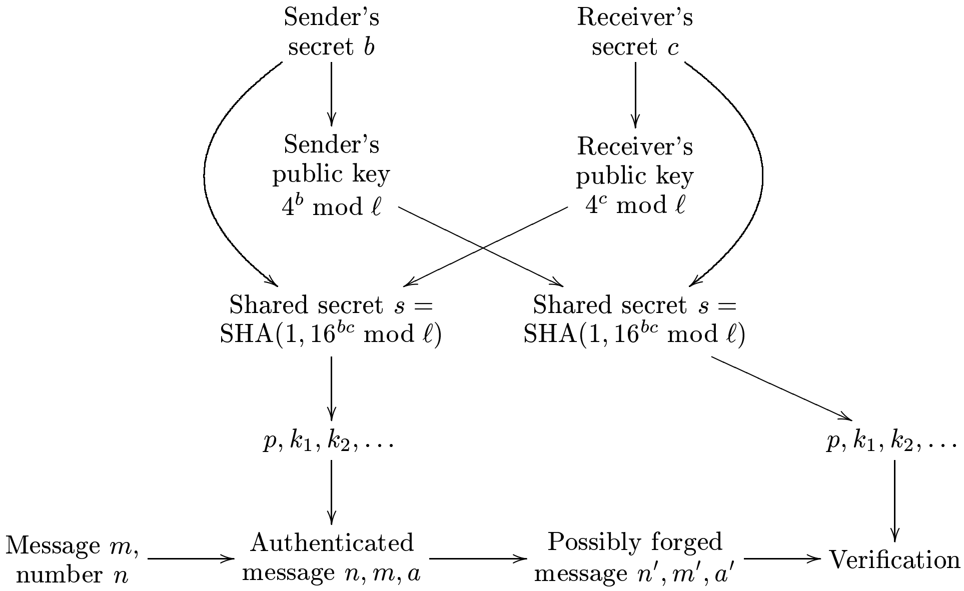
Similarly, the receiver starts from a secret uniform random integer c with $0 \le c < 2^{256}$. The receiver computes and announces a public key $4^c \bmod \ell$.

The sender now computes $16^{bc} \bmod \ell$ as the $2b$th power of the receiver's public key $4^c \bmod \ell$. The receiver computes $16^{bc} \bmod \ell$ as the $2c$th power of the sender's public key $4^b \bmod \ell$.

Finally, the sender and receiver compute $s = \mathrm{SHA}(1, 16^{bc} \bmod \ell)$. SHA is the function defined in [3], producing 256-bit output. The definition of SHA, like the definition of MD5, is much too complicated to be repeated here.

It *appears* to be very difficult for an attacker to distinguish the shared secret s from a uniform random 256-bit string. The attacker is given $4^b \bmod \ell$ and $4^c \bmod \ell$, but neither b nor c. The fastest known method of figuring anything out about s is to use the number field sieve to compute b from $4^b \bmod \ell$. This computation is not feasible today.

The sender and receiver can use this shared secret s to protect the authenticity of a series of messages:



There is no fast method *known* for a forger to take advantage of the public keys $4^b \bmod \ell$ and $4^c \bmod \ell$.

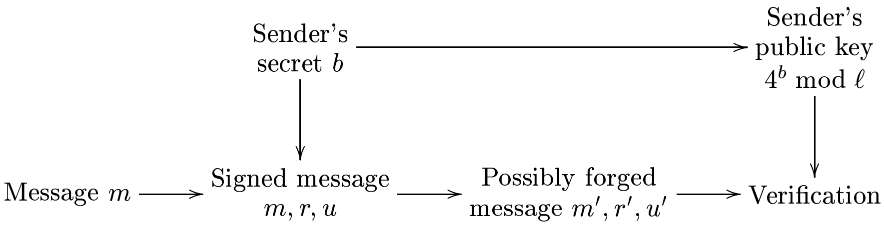
History. Diffie and Hellman in [13] introduced the general idea of sharing a secret through a public channel, and the specific approach shown above.

There are several popular variants of the Diffie-Hellman system, using groups other than $(\mathbf{Z}/\ell)^*$. One interesting variant uses the group of points on an elliptic curve over $\mathbf{Z}/\ell$ for ℓ around 2^{256}. This allows much shorter public keys, only 256 bits instead of 1536 bits, and is still conjectured to be safe. See the book [10] for more information on elliptic-curve cryptography.

See [23] for an introduction to modern discrete-logarithm methods. See [30] for a recent example of what can be done with the number field sieve. See my papers [7] and [8] for asymptotic improvements; [7] includes a comprehensive bibliography.

5. Public-key signatures

Here is a protocol for the sender to protect a message sent to many receivers:



The sender starts from a secret uniform random integer b with $0 \le b < 2^{256}$. As in the previous section, the sender computes and announces a public key $4^b \bmod \ell$.

Given a message m, the sender selects a secret uniform random integer e with $0 \le e < 2^{256}$; computes $r = 4^e \bmod \ell$; and finds an integer u with $0 < u < (\ell-1)/2$ such that $e \equiv u(\mathrm{SHA}(2,m) + \mathrm{SHA}(3,r)b) \pmod{(\ell-1)/2}$. If no such integer u exists, the sender tries a new e. The sender then transmits (m, r, u).

The receiver discards (m', r', u') unless $r' = 4^{\mathrm{SHA}(2,m')u'} n^{\mathrm{SHA}(3,r')u'} \bmod \ell$ and $0 < u' < (\ell-1)/2$, where $n = 4^b \bmod \ell$.

The pair (r, u) is a **signature** of m. Signatures are different from authenticators: a signature can be verified by anyone, while an authenticator can be verified only by people who could have created the authenticator. The receiver can convince third parties that the sender signed a message; the receiver cannot convince third parties that the sender authenticated a message. Signatures are appropriate for public communications; authenticators are appropriate for private communications.

It *seems* that forging signatures is very difficult. As in the previous section, the fastest known attack is to use the number field sieve to compute b from $4^b \bmod \ell$. This computation is not feasible today.

History. Diffie and Hellman in [13] introduced the idea of public-key signatures, but did not have any useful examples. Rivest, Shamir, and Adleman in [26] are often credited with the first example; but the original RSA system is obviously breakable. Rabin in [24] presented a variant of RSA that appears to be unbreakable. See my paper [5] for a state-of-the-art RSA-type system.

ElGamal in [15] introduced the type of signature system shown above. Signature verification in the ElGamal system is much slower than signature verification in the Rabin system; however, elliptic-curve variants of the ElGamal system offer much smaller signatures than the Rabin system does.

References

[1] —, *23rd annual symposium on foundations of computer science*, IEEE Computer Society, New York, 1982. MR 85k:68007.

[2] —, *38th annual symposium on foundations of computer science*, IEEE Computer Society, Los Alamitos, 1997. ISBN 0-8186-8197-7.

[3] —, *Descriptions of SHA-256, SHA-384, and SHA-512*, National Institute of Standards and Technology, Washington, 2000; available from `http://csrc.nist.gov/encryption/shs/sha256-384-512.pdf`.

[4] Daniel J. Bernstein, *How to stretch random functions: the security of protected counter sums*, Journal of Cryptology **12** (1999), 185–192; available from `http://cr.yp.to/papers.html#stretch`. MR 2000b:94015.

[5] Daniel J. Bernstein, *A secure public-key signature system with extremely fast verification*, to appear, Journal of Cryptology; available from `http://cr.yp.to/papers.html#sigs`.

[6] Daniel J. Bernstein, *Floating-point arithmetic and message authentication*, submitted for publication; available from `http://cr.yp.to/papers.html#hash127`.

[7] Daniel J. Bernstein, *How to find small factors of integers*, submitted for publication; available from `http://cr.yp.to/papers.html#sf`.

[8] Daniel J. Bernstein, *The cost of integer factorization*, draft.

[9] John Black, Shai Halevi, Hugo Krawczyk, Ted Krovetz, Phillip Rogaway, *UMAC: fast and secure message authentication*, in [32] (1999), 216–233; available from `http://www.cs.ucdavis.edu/~rogaway/papers/umac.html`. MR 1 729 300.

[10] Ian F. Blake, Gadiel Seroussi, Nigel P. Smart, *Elliptic curves in cryptography*, Cambridge University Press, Cambridge, 2000. ISBN 0–521–65374–6. MR 1 771 549.

[11] G. R. Blakley, David Chaum (editors), *Advances in cryptology: CRYPTO '84*, Lecture Notes in Computer Science 196, Springer-Verlag, Berlin, 1985. ISBN 3–540–15658–5. MR 86j:94003.

[12] Lenore Blum, Manuel Blum, Michael Shub, *A simple unpredictable pseudorandom number generator*, SIAM Journal on Computing **15** (1986), 364–383. MR 87k:65007.

[13] Whitfield Diffie, Martin Hellman, *New directions in cryptography*, IEEE Transactions on Information Theory **22** (1976), 644–654. MR 55 #10141.

[14] Taher ElGamal, *A public key cryptosystem and a signature scheme based on discrete logarithms*, in [11] (1985), 10–18. MR 87b:94037.

[15] Taher ElGamal, *A public key cryptosystem and a signature scheme based on discrete logarithms* **31** (1985), 469–472; draft in [14]. MR 86j:94045.

[16] Edgar N. Gilbert, F. Jessie MacWilliams, Neil J. A. Sloane, *Codes which detect deception*, Bell System Technical Journal **53** (1974), 405–424. MR 55 #5306.

[17] Oded Goldreich, *Modern cryptography, probabilistic proofs and pseudorandomness*, Springer-Verlag, Berlin, 1999. ISBN 3–540–64766–X. MR 2000f:94029.

[18] Hugo Krawczyk (editor), *Advances in cryptology: CRYPTO '98*, Lecture Notes in Computer Science 1462, Springer-Verlag, Berlin, 1998. ISBN 3–540–64892–5. MR 99i:94059.

[19] Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone, *Handbook of applied cryptography*, CRC Press, Boca Raton, Florida, 1996; available from `http://cacr.math.uwaterloo.ca/hac/index.html`. ISBN 0–8493–8523–7. MR 99g:94015.

[20] Gary L. Mullen, Peter Jau-Shyong Shiue (editors), *Finite fields: theory, applications, and algorithms*, American Mathematical Society, Providence, 1994. ISBN 0–8218–5183–7. MR 95c:11002.

[21] Moni Naor, Omer Reingold, *Number-theoretic constructions of efficient pseudo-random functions*, in [2] (1997), 458–467.

[22] Wim Nevelsteen, Bart Preneel, *Software performance of universal hash functions*, in [28] (1999), 24–41.

[23] Andrew M. Odlyzko, *Discrete logarithms and smooth polynomials*, in [20] (1994), 269–278. MR 95f:11107.

[24] Michael O. Rabin, *Digitalized signatures and public-key functions as intractable as factorization*, Technical Report 212, MIT Laboratory for Computer Science, 1979; available from `http://hdl.handle.net/ncstrl.mit_lcs/MIT/LCS/TR-212`.

[25] Ronald L. Rivest, *The MD5 message-digest algorithm*, Request For Comments 1321 (1992); available from `http://theory.lcs.mit.edu/~rivest/rfc1321.txt`.

[26] Ronald L. Rivest, Adi Shamir, Leonard M. Adleman, *A method for obtaining digital signatures and public-key cryptosystems*, Communications of the ACM **21** (1978), 120–126.

[27] Bruce Schneier, *Applied cryptography: protocols, algorithms, and source code in C*, 2nd edition, Wiley, New York, 1996. ISBN 0–471–12845–7.

[28] Jacques Stern (editor), *Advances in cryptology—EUROCRYPT '99*, Lecture Notes in Computer Science 1592, Springer-Verlag, Berlin, 1999. ISBN 3–540–65889–0. MR 2000i:94001.

[29] Alan M. Turing, *Computing machinery and intelligence*, MIND **59** (1950), 433–460. MR 12,208c.

[30] Damian Weber, Thomas Denny, *The solution of McCurley's discrete log challenge*, in [18] (1998), 458–471. MR 99i:94057.

[31] Mark N. Wegman, J. Lawrence Carter, *New hash functions and their use in authentication and set equality*, Journal of Computer and System Sciences **22** (1981), 265–279. MR 82i:68017.

[32] Michael Wiener (editor), *Advances in cryptology—CRYPTO '99*, Lecture Notes in Computer Science 1666, Springer-Verlag, Berlin, 1999. ISBN 3-540-66347-9. MR 2000h:94003.

[33] Andrew C. Yao, *Theory and applications of trapdoor functions*, in [1] (1982), 80–91.

Department of Mathematics, Statistics, and Computer Science (M/C 249), The University of Illinois at Chicago, Chicago, IL 60607–7045

E-mail address: `djb@cr.yp.to`