Cryptography: Authentication, Blind Signatures, and Digital Cash*

Rebecca Bellovin

1 Introduction

One of the most exciting ideas in cryptography in the past few decades, with the widest array of applications, is asymmetric encryption, or public key cryptography. Asymmetric encryption is the idea that an encryption scheme can be broken up into two parts, so that anyone can encrypt, but only someone who knows the secret key can decrypt. That contrasts with traditional cryptography, where the same key is required for encryption and decryption.

The most famous example of public key cryptography is the RSA encryption scheme, invented by Ron Rivest, Adi Shamir, and Len Adleman. RSA is based on the idea that factoring large numbers is hard, even though multiplication is easy. To make a public key, Alice first picks two large primes, p and q, and multiplies them together to get n. She picks some $d \in U_{\varphi(n)}$, which she keeps secret, and computes $e \in U_{\varphi(n)}$ such that $ed \equiv 1 \pmod{\varphi(n)}$, which she publishes, along with the modulus n. Then to encrypt a message $m \in U_n$ for Alice, Bob computes the ciphertext $c := m^e \pmod{n}$. To read Bob's message, Alice computes

$$c^d = (m^e)^d = m^{ed} = m^{1+\varphi(n)k} \equiv m \pmod{n}$$

This works because $m^{\varphi(n)} \equiv 1 \pmod{n}$ for all $m \in U_n$, as you know from the number theory problem sets. And no one but Alice knows the decryption key d, because she didn't publish it and computing it from e seems to require knowledge of $\varphi(n)$, which in turn seems to require knowledge of p and q.

Another common encryption scheme is Rabin encryption, which is based on the idea that extracting square roots modulo n is hard, unless you know how to factor n. Again, Alice chooses two large secret primes, p and q, and multiplies them together to get n, which she publishes. Generally, p and q are taken to be 3 modulo 4, but this is not a necessity. Then

*The blind signature schemes and applications I discuss are primarily the work of D. Chaum, beginning with a paper "Blind Signatures for untraceable payments" (Advances in Cryptology—Crypto '82), which was followed by a joint paper "Untraceable electronic cash" (Proceedings on Advances in Cryptology) with A. Fiat and M. Naor in 1990.

to encrypt a message $m \in U_n$, Bob computes $c := m^2 \pmod{n}$. To decrypt, Alice computes the square roots of c in U_p and U_q, and then uses the Chinese remainder theorem to find four candidates for m in U_n. Alice's computations are simple, because if c is a quadratic residue modulo p for $p \equiv 3 \pmod{4}$, then $c^{\frac{p+1}{4}} \equiv c \cdot c^{\frac{p-1}{2}} \equiv c \pmod{4}$, by Euler's criterion.

In addition to hiding data, public key cryptography is can be used for authentication. Imagine the following scenario: Alice sends a message to Bob, encrypted with Bob's public key, but Mal, the malicious attacker, is sitting in the middle. Mal intercepts Alice's message, which tells Bob to send her money, and in its place sends a message purporting to be from Alice, telling Bob to send the money to him. Bob has no way of knowing that the message he receives did not really come from Alice. So we want a way for Alice to assure Bob that the message he gets is really from her.

What Alice can do is sign her message. That is, before encrypting her message M with Bob's public key E_b, she applies a hash function f (a function that is computationally hard to invert) to M and decrypts $f(M)$ with her own private key D_a. So she sends Bob the pair $(E_b(M)), D_a(f(M)))$. When Bob gets it, he applies his private key D_b to the message part, Alice's public key E_a to the second part, and checks to make sure $f(D_b(E_b(M))) = E_a(D_a(f(M)))$. Since we assume Mal does not know Alice's private key and cannot invert f, Bob is now certain the message came from Alice: while Mal could have replaced the message, he could not have produced something that, when encrypted with Alice's public key, gives a message with chosen hash value.

2 Blind Signatures and Voting

One of the key features of the signing algorithm discussed in the last section is that Alice knows exactly what she is signing—she is signing it because she is the author of the message. But the idea of authentication applies to more situations that simply proving one is the author of a particular message. For example, let us consider an election. A voter who presents identification is actually authenticating two things: his or her identity, and the validity of his or her vote. A voter stepping into a voting booth takes care of both steps at once, but a person voting by mail or electronically could have the secrecy of his or her vote compromised. After all, if a ballot is mailed in an envelope with a return address, anyone processing the vote could simply note who it came from (assuming the US postal service will not mail letters with incorrect return addresses, of course). But if the election board permitted ballots to be mailed in unmarked envelopes, the potential for fraud rises. And if an election is conducted online, the link between a voter and his or her vote is stronger, and the potential for fraud in anonymous elections is much higher.

Instead, we can imagine a protocol for voting that separates these authentication steps. In this scenario, Alice is a trustee, and any ballot she certifies valid will be counted. To vote, Bob first puts his ballot in a carbon paper-lined envelope[1], seals it, and then mails that

[1]If you never played with carbon paper as a child, the important thing is that if put carbon paper over

to Alice in an outer envelope with his return address on it. Alice checks the voter rolls to make sure Bob is eligible to vote, takes out the inner envelope and signs it (without opening it), then mails it back to Bob (since she has his return address). Bob takes out his ballot and mails that anonymously to the vote counting center, and since the ballot has Alice's signature on it, his vote is counted. And since Alice never saw his ballot, and the vote counters never saw his name, no one knows how he voted.

Let us review the ideas and assumptions that make this work:

- Bob can prove to Alice that he is who he says he is, and therefore has the right to vote.
- No one can duplicate Alice's signature.
- Anyone can check that Alice's signature is valid.
- Bob can seal his carbon paper envelope in such a way that Alice is incapable of opening it.
- The only difference between the ballot Bob sent to Alice and the ballot he mailed to the vote counters is Alice's signature.

Now we can try to translate this to the language of cryptography and electronic transactions. The first condition is simply the sort of authentication scheme we have already discussed. The second and third conditions require that Alice have a good signature scheme S' with a public inverse S. The fourth condition says that Bob has an encryption scheme E_b, with inverse D_b known only to him. The last condition tells us that E_b and S' are compatible, in the sense that $D_b(S'(E_b(M))) = S'(M)$ for any message M. In short, if appropriate encryption and signature functions exist, we have the following protocol:

1. Bob applies E_b to his ballot B, and sends $E_b(B)$ to Alice, using some sort of authentication scheme.
2. Alice verifies Bob's authentication, applies S' to $E_b(B)$, and sends $S'(E_b(B))$ back to Bob.
3. Bob applies D_b to $S'(E_b(B))$ to get $S'(B)$, and checks to make sure $S(S'(B)) = B$. This ensures that Mal has not replaced his ballot.
4. Bob sends $S'(M)$ to the vote counters, anonymously.

To prevent multiple voting, the election board can require that ballots have a random component followed by the actual vote, so all the possible values $S'(B)$ should be different. Then if the vote counters see multiple copies of a ballot, they know someone tried to commit vote

ordinary paper and write on the carbon paper, the writing appears on the paper beneath.

fraud and only count one (with sufficient redundancy in the encryption scheme, arrangements like this are vanishingly unlikely to disenfranchise legitimate voters).

As a matter of fact, appropriate encryption and signature schemes exist; Paillier encryption, which Jeremy talked about, is generally used. Thus, we have used the power of blind signatures to separate the authentication of Bob's identity from the authentication of his ballot.

3 Digital Cash

The same sort of idea can be used to construct a digital analogue of cash. The key property of cash is anonymity: when you take money out of the bank, the bank gives you the cash without knowing what you buy, and when you spend money, the merchant has no idea who you are. By contrast, when you buy something with a credit card online, you have to tell the merchant who you are, and you have to tell the credit card company who you are making a purchase from. The potential for invasion of privacy is immense.

For the purposes of this construction, we will assume that all coins are worth a dollar. To withdraw a dollar from her account, Alice generates a coin C, applies a public hash function f, and masks the result by encrypting it with E_a. The bank signs $E_a(f(C))$ with S' and debits Alice's bank account. Alice then computes $D_a(S'(E_a(f(C))))$ to strip away her encryption, leaving her with $S'(f(C))$, and checks to make sure $S(S'(f(C))) = f(C)$. To spend her dollar, Alice gives $S'(f(C))$ and C to a merchant. The merchant computes $S(S'(f(C)))$ and compares that to $f(C)$ to make sure the coin was actually signed by the bank (without the use of f, Alice could simply have taken a random X and presented the pair $(X, S(X))$ as a pair $(S'(C), C)$). Then the merchant sends $S'(f(C))$ and C to the bank, which checks the validity of the signature, pays the merchant, and puts C on a list of coins that have already been spent.

This scheme preserves Alice's anonymity and it lets the bank detect double-spending, but it provides no way to punish double-spenders. The only way Alice could be caught trying to double-spend is if the merchant is online and has the bank check her coin in real time.

Jeremy talked about zero-knowledge proofs, which allow Alice to prove to Bob that she knows something, for example a square root modulo a composite n, while not revealing the information itself. David Chaum, Amos Fiat, and Moni Naor used similar ideas to build a digital cash protocol that lets Alice maintain her anonymity so long as she doesn't cheat; the penalty for cheating is having her identity revealed.

To begin with, the bank fixes a security parameter k that determines how likely it is to catch double-spenders—large k make it harder to cheat, and the probability that a cheater will be caught goes to 1 very quickly as k increases. Additionally, Alice has an account number u with an associated counter v (both Alice and the bank know the account number and the counter), and there are two functions $f, g : \mathbb{Z}_n \times \mathbb{Z}_n \to \mathbb{Z}_n$ which are hard to invert. One of

Alice's coins consists of k 4-tuples $(a_i, c_i, d_i, r_i) \in \mathbb{Z}_n^4$, which she chooses at random. To get a coin signed, Alice proceeds as follows:

1. Alice computes the k blinded values $B_i = E_a(f(x_i, y_i))$, where $x_i = g(a_i, c_i)$ and $y_i = g(a_i \oplus (u \parallel (v+i)), d_i)$, and sends them to the bank. Here $\oplus$ denotes bitwise exclusive or, and $\parallel$ denotes concatenation. Alice uses r_i as a key for E_a, one for each candidate B_i.

2. The bank chooses a random set R of $k/2$ indices, and sends it to Alice. Alice then reveals (a_i, c_i, d_i, r_i) for $i \in R$. The bank can check that these 4-tuples yield the values of B_i Alice claimed (since it knows u and v), so if Alice tried to cheat, it is likely that the bank catches her at this stage.

3. The bank sends Alice the signed masked coin $S'(\prod_{i \notin R} B_i)$, debits her account, and increments the counter v by k. Here $\prod$ refers to some operation preserved by Alice's encryption scheme and the signing scheme; in the original paper it is actually multiplication, because the signing scheme was extraction of cube roots and Alice's encryption was multiplication by r_i^3. Alice now applies D_a to find the value $C = S'(\prod_{i \notin R} f(x_i, y_i))$, and she can check that this is correct by applying S, so she increments v by k.

Now that Alice has her electronic coin worth a dollar, she can pay Bob with it:

1. Alice gives C to Bob.

2. Bob sends Alice a random binary string of length $k/2$; if the ith bit is 1, then Alice sends Bob a_i, c_i, and y_i, and if the ith bit is 0, then Alice sends x_i, $a_i \oplus (u \parallel (v+i))$ and d_i. Bob can now check that these pieces of data fit the value of C Alice provided. Since f and g are assumed to be impossible to invert, and the bank's signing scheme is assumed to be impossible to duplicate, if Alice is lying she is likely to be caught.

3. Bob sends the transcript of his conversation to the bank, which pays him and keeps the transcript on file.

Note that Alice is anonymous in her interaction with Bob; the only identifying information she gives him is her account number, but that is XOR'ed with a random string a_i, so Bob cannot see it. The numbers c_i and d_i are there so that even if g is a weaker hash function than expected, anyone who breaks it still has an enormous number of possible pairs (a_i, c_i) and $(a_i \oplus (u \parallel (v+i)), d_i)$ which g maps to x_i and y_i. So even if Bob can invert g he does not know who Alice is. But consider what happens if Alice tries to spend the same coin twice: the bank will compare the transcripts from two purchases where she used the coin. Because of the randomized challenge strings, it is very likely that there is some i such that the ith bit of the random challenge string was 0 in one interaction and 1 in the other. Then the bank can XOR her two responses to get $u \parallel (v+i)$, which tells it Alice's account number and which withdrawal she double-spent.

One problem with this scheme as written is that while the bank can figure out that Alice tried to defraud them, the bank can also forge transcripts to frame her. Thus, the bank cannot prove to anyone else that Alice was guilty. But assuming Alice has a digital signature scheme of her own, it is easy to modify the above protocol to protect her. When Alice is choosing her 4-tuples, she randomly chooses another k pairs of integers (z_i', z_i''), and signs the concatenation $g(z_1', z_1'') \parallel \cdots \parallel (z_k', z_k'')$. She gives that signed number to the bank. Additionally, when she is computing the values $f(x_i, y_i)$, she replaces u with $u \parallel z_i' \parallel z_i''$ (and provides the appropriate values to the bank during the random verification phase). Thus, if the bank can provide $k/2 + 1$ correct pairs (z_i', z_i''), it has proof that Alice cheated. The bank cannot change the given values of $g(z_i', z_i'')$ because it cannot break Alice's signature. Even if we assume the bank can invert g, Alice only has to display her own pair (z_i', z_i'') to show that g has actually been broken.

Thus, cryptography has given us a protocol for spending money online so that that money functions like cash does in the physical world. On an aesthetic level, this construction could be better. The generation of money is not truly zero-knowledge, because Alice overgenerates data and reveals some of it selectively to demonstrate that it is all correctly done. The compromised data is then thrown away, but it seems likely to me that a zero-knowledge verification protocol exists. On a more practical level, the construction I have given only allows for a single denomination. As not all prices are in dollar multiples, it seems that the number of transactions required, and the associated computational power and storage space, would be prohibitive. There are variants on this protocol that provide for digital cash in multiple denominations, though. Actually, the practicality is largely moot at this point because there is no infrastructure using digital cash. Chaum founded DigiCash to try to implement it, but the company went bankrupt some time ago.