Menu

A Few Thoughts on Cryptographic Engineering

Some random thoughts about crypto. Notes from a course I teach. Pictures of my dachshunds.

A note on blind signature schemes

RSA blind signatures

Traditional RSA signatures have the form $S = M^d \bmod N$, where M is the message, (N, e) is the public key, and d is the secret key, selected such that for any m: $m^{e*d} = m$ (see here (http://en.wikipedia.org/wiki/RSA_(algorithm))for details on how keys are constructed). Chaum observed that a user could 'blind' an RSA message for a bank to sign, by first selecting a random r (in the range 1 through N-1, such that r has an inverse $\bmod N$) and giving the bank the blinded value ($M * r^e$) mod N.

The bank can compute $(M * r^e)^d \bmod N$ using its secret key, and return this 'almost signed' value to the user. Since d is selected to such that any $m^{ed} = m$, the equation simplifies to:

$$(M * r^e)^d \equiv (M^d * r^{ed}) \equiv M^d * r \qquad \bmod N$$

The user can simply divide out r (technically: multiply by $r^{-1} \bmod N$) to obtain the actual signature $M^d \bmod N$.

(Side note: many crypto libraries employ this technique for a different purpose — to avoid timing attacks on RSA decryption. By 'blinding' the value before applying the secret key, the library prevents the attacker from submitting specific numbers to be decrypted. This stops a class of known remote timing attacks (http://crypto.stanford.edu/~dabo/papers/ssl-timing.pdf).)

Schnorr/DSA blind signatures

The DSA scheme is based (not loosely) on the Elgamal and Schnorr signature schemes. Let (g, p, q, y) be the public key, where (g, p, q) describe a group of order q, with generator g (see here (http://en.wikipedia.org/wiki/Schnorr_signature) for details on how these elements are chosen) and $y = g^x \bmod p$. Let H() be a hash function that maps to elements in the space $(1, 2, \ldots, q-1)$.

To form a traditional (non-blind) Schnorr signature on the message M, the signer picks a random k between (1 and q-1) and computes the signature (r, s) as:

$$r = g^k \bmod p, \quad s = (H(M \,||\, g^k)*x + k) \bmod q$$

The symbol || indicates concatenation. Any party can verify this signature using the public key, by checking the following equality:

$$g^s == y^{H(M \,||\, r)} * g^k \quad \bmod p$$

To turn the above into a *blind signature*, the user and the bank engage in the following protocol.

1. The bank picks k as above, and generates $r = g^k \bmod p$. It sends r to the user.
2. The user now picks random a, b in (1, … q-1) and computes (r', e', e) as:
   $r' = r*(g^a)*(y^b) \bmod p$
   $e' = H(M \,||\, r')$
   $e = e' - b \bmod q$

   She sends e to the bank and retains the rest.

3. The bank computes $s = ex + k$ mod q and returns s.
4. The user computes $s' = s + a \bmod q$. The pair (r', s') is a valid signature on M.

Whew! Ok, note that the bank sees only the value e, which is based on a hash of M, but is *blinded* by the random factor b. This ensures that the bank does not learn anything about M.
The intuition for correctness is a little bit more complicated, and I'm going to leave it for the reader to work out. But it too holds: the new pair (r', s') is also a valid solution to the signature verification equation described above.
Blind DSA signatures are quite a bit more complicated, but can also be accomplished. See this paper (http://www.ece.cmu.edu/~reiter/papers/2001/CRYPTO.pdf) for a protocol.

Are there other blind signature schemes?

Tons. My favorite is the Camenisch-Lysyanskaya (http://www.zurich.ibm.com/~jca/papers/camlys02b.pdf) class of signatures, which allow you to extract signatures on message vectors — lists of messages — then show the signature on any subset at all. This signature scheme is the basis of many anonymous credential (http://en.wikipedia.org/wiki/Digital_credential#Anonymous) schemes, as well as some eCash protocols. There are also many more recent blind signatures in newer settings, such as bilinear groups (http://courses.csail.mit.edu/6.897/spring04/L25.pdf).

Selectively revealing a cheater's identity

The problem of revealing a user's identity (on double spend) is kind of an interesting one.

The first proposal actually comes from Chaum, Fiat and Naor (http://blog.koehntopp.de/uploads/chaum_fiat_naor_ecash.pdf), but that one's a little complicated. For discussion purposes, it's probably easier to talk about this later scheme (http://courses.csail.mit.edu/6.857/2009/handouts/untraceable.pdf) by Stefan Brands.

Brands' scheme is neat because it actually turns a known vulnerability of signature schemes like Schnorr and DSA into an asset that prevents double-spending.

You might remember I mentioned in a previous post (https://blog.cryptographyengineering.com/2012/03/surviving-bad-rng.html) that there's a serious concern in signature schemes like DSA. These schemes use a random *nonce* value, and if that nonce is ever re-used twice (with two different messages), anyone can recover the signer's key (http://rdist.root.org/2010/11/19/dsa-requirements-for-random-k-value/). Brands' scheme actually turns this weakness into a feature: speaking at a very high level, each coin withdrawn from the bank consists of a bank-signed secret value and a single secret nonce (broken into pieces).

Spending a Brands coin is something like making a signature using this key and nonce; the user can do it once with no worries, but the *second* time she does it, anyone can recover her key.

This is a pretty rough, inaccurate intuition. For a much more detailed explanation of Brands' scheme, see here (http://www.orlingrabbe.com/stefbrdc.htm).

2 thoughts on "A note on blind signature schemes"

Haily Barrington says:
November 1, 2012 at 5:09 am
This is really technical and it's pretty rough too. But you really made it simple to understand and it's very accurate. Thanks for sharing this one.

Anonymous says:
March 19, 2013 at 8:20 am
Thanks a lot for a concise and yet pretty informative note on blind signatures!

Comments are closed.

Menu