Hashing and Digital Signatures Notes
Aaron Blumenfeld

ElGamal: Public α, p. Choose a secret a and make $\beta \equiv \alpha^a$ public. To send a message, choose random k and send (r, t), where $r \equiv \alpha^k, t \equiv \beta^k m$. Decryption: $tr^{-a} \equiv m \pmod{p}$.

ElGamal Signatures: Public: α, p, β, where $\beta \equiv \alpha^a$, secret a. To sign m, choose random k with $(k, p-1) = 1$. Let $r \equiv \alpha^k \pmod{p}, s \equiv k^{-1}(m - ar) \pmod{p-1}$. The signed message is (m, r, s). To verify, let $v_1 \equiv \beta^r r^s \pmod{p}, v_2 \equiv \alpha^m \pmod{p}$. Valid iff $v_1 \equiv v_2 \pmod{p}$.

Hash Functions: Want $h(m)$ easy to compute, preimage resistant (hard to find any preimage), and strongly collision-free (hard to find m, m' with $h(m) = h(m')$). Weakly collision-free: given an m, hard to find m' with $h(m') = h(m)$.

Birthday Attack: Given N objects, if r people choose an object (with repetition), the probability that two people choose the same object is roughly $1 - e^{-r^2/2N}$. For $r^2/2N = \log 2$, we have $r \approx 1.77\sqrt{N}$, and the probability is 50% of having a match.

In cryptography setups, we have two rooms, each with r people choosing (with repetition) among N objects. The probability of a match in room 1 and room 2 is roughly $1 - e^{-r^2/N}$. The probability of i matches is $(r^2/N)^i e^{-r^2/N}/i!$. Taking $r = \sqrt{N}$ gives us a good chance of finding a match. If need, can change r to be something like $5\sqrt{N}$. This can be used in a way similar to Baby Step Giant Step for computing discrete logs.

If a hash function spits out a small output (n-bits). There are $N = 2^n$ possible outputs, take $r = 2^{n/2+k}$ for some small k to find a collision. Find $n/2 + k$ places to change spacing, commas, etc. in a fraudulent document and hence make $2^{n/2+k}$ different documents. Also make $2^{n/2+k}$ different legitimate documents. Make two lists of length $2^{n/2+k}$ of the hashes and find a match using the birthday paradox. The probability of finding a match is $1 - \exp(2^{2k}) \approx 1$ even for $k = 3$. Then have Alice sign the corresponding legitimate document and append her signature to the fraudulent document.

Blind Signatures: (For RSA). To get A to sign m, choose a random r with $(r, n) = 1$ and calculate $m' \equiv mr^e \pmod{n}$. Since r is random, so is r^e ($r \mapsto r^e$ is a one-to-one map). Hence m' is quite likely meaningless. So have Alice sign m', giving you $s' \equiv m'^d \equiv (mr^e)^d \equiv rm^d$. Then divide by r, and you've got Alice's signature for your message m.

The Digital Signature Algorithm: Assume for simplicity signing a 160-bit message. Alice chooses q prime that is 160 bits long with $q \mid p - 1$. Discrete logs should be hard for this p. Let g be a primitive root $\pmod{p}$, put $\alpha \equiv g^{(p-1)/q}$. Alice chooses a secret $a \pmod{q}$ and computes $\beta \equiv \alpha^a \pmod{p}$. (p, q, α, β) is public, a is secret.

Alice signs m: chooses random $k \pmod{q}$, calculates $r \equiv \alpha^k \pmod{p} \pmod{q}$, $s \equiv k^{-1}(m + ar) \pmod{q}$. She sends (r, s) to Bob, along with m. *Bob verifies*: calculates $u_1 \equiv s^{-1}m \pmod{q}, u_2 \equiv s^{-1}r \pmod{q}, v \equiv \alpha^{u_1}\beta^{u_2} \pmod{p} \pmod{q}$. Signature is valid iff $v \equiv r$.

Chapter 10: Security Protocols: To avoid intruder-in-the-middle attacks, need *certification* and *validation*. Certification binds a public key to some entity (e.g., Bob $\leftrightarrow (e_B, n_B)$). Validation guarantees the certificate is valid. Certification Authorities (CAs): Verisign (57%), Comodo (8.3%), GoDaddy (6%), etc.

CAs are assumed to be trustworthy. They produce their own certificate and sign it. It is posted on website (sometimes packaged with browsers). For a fee, CAs will produce a certificate for clients (possibly smaller companies). Here, trust is critical. Different CAs trust each other, so Alice and Bob having different CAs should be a nonissue. RAs (registration authorities) are often authorized by CAs to sign certificates; fairly complex hierarchy networks of internet trust are thus established.

For example, if you attempt to view the certificates on the WeBWorK page, you will see "Could not verify this certificate for unknown reasons." If you're still taking Calculus classes, you can blame Eve for diverting your answer submissions when asking for an extension.