Computer Science Tripos part 2 – Security

Lecture 10 – Public Key Algorithms

Ross J. Anderson

In the 1b course, we spent one lecture giving a brief overview of public key cryptography. We also included the famous 'New Directions' paper by Diffie and Hellman which introduced the world to the ideas of public key encryption and digital signature in 1976, and Ellis's 'History of Non-secret Encryption' which describes his independent discovery of public key encryption at GCHQ in 1970. I'd suggest you re-read these papers.

In the next two lectures, we will look at public key cryptology and the protocols in which it is used in somewhat greater detail.

1 Revision: Number Theory

Recall that the prime numbers are the positive whole numbers with no proper divisors. By definition, 1 is not prime; so the primes are $\{2, 3, 5, 7, 11, ...\}$. **The fundamental theorem of arithmetic** states that each natural number greater than 1 factors into prime numbers in a way which is unique up to the order of the factors. There are infinitely many primes, as if there were only k primes 2,3,5, ..., p_k, then the number $q = 2.3.5...p_k + 1$ would be either a prime, or divisible by a prime greater than p_k, which is a contradiction. Indeed, if we write $\pi(x)$ for the number of prime numbers up to and including x (so, for example, $\pi(10) = 4$ (2,3,5,7)), the Prime Number Theorem states that

$$\pi(x) \sim \frac{x}{\log x}$$

So the probability of a random M-bit number being prime is about $1/\log M$. This enables us to estimate the running time of algorithms which search for large prime numbers by draw-and-test.

The Chinese Remainder Theorem states that if p and q have no common factor, then any congruence modulo pq is specified by its values mod p and mod q – we can solve $x \equiv a \pmod p$ and $x \equiv b \pmod q$ as $x = aq + bp$, and all other solutions are congruent to this modulo pq. This theorem is used to speed up arithmetic modulo pq, while the polynomial version of it can be used to show that breaking two cyclic redundancy checks (as used in the McAfee anti-virus toolkit) is no harder than breaking one large one.

Fermat's (little) theorem states that for all primes p not dividing a, $a^{p-1} \equiv 1 \pmod p$ (proof: take the set $\{1, 2, ..., p-1\}$ and multiply each of them modulo p by a, then cancel out $(p-1)!$ each side). Euler's function $\phi(n)$ is the number of positive integers less than n with which it has no divisor in common; so if n is the product of two primes pq then $\phi(n) = (p-1)(q-1)$ (proof similar).

Discrete Logarithms: A primitive root modulo p is a generator of the multiplicative group modulo p, or in less highbrow language a number whose powers generate all the nonzero numbers mod p. For example, 5 is a primitive root modulo 7:

$$\begin{array}{llll}
5^1 & & = 5 & \pmod 7 \\
5^2 = & 25 & \equiv 4 & \pmod 7 \\
5^3 \equiv & 4.5 & \equiv 6 & \pmod 7 \\
5^4 \equiv & 6.5 & \equiv 2 & \pmod 7 \\
5^5 \equiv & 2.5 & \equiv 3 & \pmod 7 \\
5^6 \equiv & 3.5 & \equiv 1 & \pmod 7
\end{array}$$

This means that given any y, we can always solve the equation $y = 5^x \pmod 7$; x is then called the discrete logarithm of y modulo 7 to the base 5. Observe in passing that 5^2 generates a subgroup of order 3, and 5^3 generates a subgroup of order 2.

2 Public Key Cryptography

The idea of Ellis and of Diffie and Hellman was that an encryption algorithm need not encrypt and decrypt with the same key. If it were possible to arrange things so that an opponent who was given your encryption key could not work out the decryption key, then the following solution to the key management problem would be possible.

Let the encryption (public) key be K, and the decryption (private) key be K^{-1}. A public key algorithm should have the property that although anybody can generate a pair of keys K_X, K_X^{-1}, our opponent Charlie cannot deduce K_X^{-1} from K_X unless he was the person who generated these keys (or the value of K_X^{-1} leaked).

Thus Alice can publish her encryption key K_A, and when Bob wants to communicate with her he can simply look it up in the directory and encrypt his message M with it:

$$C = \{M\}_{K_A}$$

Now Alice alone knows K_A^{-1}, so she alone can recover the message by calculating $M = \{C\}_{K_A^{-1}}$.

The obvious problem is that Charlie might generate a keypair K_X, K_X^{-1} and place K_X in the directory with the legend "This is Alice's public key: K_X". So

Alice will want to use a directory which Charlie cannot manipulate. In other words, there is still a security requirement on key management, but we have reduced it from confidentiality to authenticity.

Can we use public key techniques to help with authenticity too? What Diffie and Hellman realised (but the GCHQ team appear not to) was that one can also have digital signatures. If our encryption and decryption operations are left and right inverses of each other ($C = \{M\}_K$ holds if and only if $M = \{C\}_{K^{-1}}$), Alice can take a message and subject it to 'decryption'

$$S = \{M\}_{K_A^{-1}}$$

which can be reversed using K_A to recover the original message

$$M = \{S\}_{K_A}$$

In this context, the 'decryption' of the message is usually referred to as a digital signature, since it can only be computed by the person possessing K_A^{-1}, while anybody who knows K_A can check it.

The consequence of this is that if Alice and Bob publish encryption keys K_A, K_B respectively while each keeps the corresponding decryption key K_A^{-1} or K_B^{-1} secret, then they can ensure both the integrity and privacy of their messages in the following elegant way. To send a message M to Bob, Alice first signs it with her private key K_A^{-1} and then encrypts it with Bob's public key K_B:

$$C = \{\{M\}_{K_A^{-1}}\}_{K_B}$$

When Bob receives this message, he can first strip off the outer encryption using his private key K_B^{-1}, and then recover the message M by applying Alice's public key K_A.

A useful analogy is that of paper correspondence. There, we first sign a letter and then put it in an envelope. Only the recipient should open the envelope, but if the letter is produced in court at some later time, then in principle anyone (e.g., the jury) can look at the signature and verify that it is that of the sender.

However, public key mechanisms have some very nasty failure modes for which an analogy with the physical world does not prepare us. For example, Charlie might try lots of random numbers R_i and encrypt them under Alice's public key until he finds one with $\{R_j\}_{K_A}$ looking like a valid message (this is called a forward search attack). He could then encrypt R_j with Bob's public key and try to pass it off to Bob. For this reason, Alice would usually want to insert redundancy into her message.

2.1 RSA

The most commonly used commercial public-key systems, such as SSL, SSH and PGP, use RSA, named after its inventors Rivest, Shamir and Adleman. Their algorithm has the above property (that encryption and decryption are both left and right inverses of each other), and so provides both public-key encryption and digital signature functions at the same time. It is based on the difficulty of factoring a large product of two primes, and the assumption that calculating cube (and other) roots modulo such a number is also hard.

The encryption key is a modulus N which is hard to factor; take $N = pq$, and choose e to have no common factors with either $p-1$ or $q-1$. The public key is (N, e) and the private key is computed from the factors p and q which are kept secret. Where M is the message and C is the ciphertext, encryption is defined by

$$C = M^e \pmod N$$

Decryption is the reverse operation:

$$M = \sqrt[e]{C} \pmod N$$

As the owner of the encryption key knows the factors p and q of N, she can easily calculate $\sqrt[e]{C} \pmod N$ as follows. Since $\phi(N) = (p-1)(q-1)$ and e has no common factors with $\phi(N)$, she can find a number d such that $de \equiv 1 \pmod{\phi N}$ using the Chinese Remainder Theorem and let

$$M = C^d \pmod N$$

This works because

$$C^d = \{M^e\}^d = M^{ed} = M^{1+k\phi(N)} = M.M^{k\phi(N)} \equiv M.1 = M \pmod N$$

Similarly, she can operate on the message with her secret key to produce a signature

$$S = M^d \pmod N$$

which anyone can verify using her public key by $S^e = M \pmod N$.

A number of warnings are appropriate here. For example, RSA is multiplicative – if we have a relation such as $M_1M_2 = M_3$ that holds among plaintexts, then the same relationship will hold among ciphertexts $C_1C_2 = C_3$ and signatures $S_1S_2 = S_3$. For this reason, it is standard practice to hash messages before signature using an algorithm such as SHA in order to break up any mathematical relationships between them and to protect against the kind of forward search attacks we discussed above.

Secondly, it is extremely dangerous to use keys for more than one purpose. For example, there are international standards for using RSA in challenge-response protocols: the principal identifying herself signs a random challenge string. If the same key is used for signature, then the opponent could present as a random challenge the hash of a message and obtain its signature. (We will discuss such 'oracle attacks' further in the next lecture.)

Thirdly, although it is a standard in some fields to use the public exponent 3 for signatures (it is a European healthcare IT standard), this is dangerous for encryption (see exercise).

Fourthly, although RSA has the property that signatures and encryption are the inverse of each other, this is dangerously deceptive for the system designer. Signature and decryption are not the same; a signed message should only have been generated by the signer, while a decrypted message could have been encrypted by anybody. The use of decryption where it was really signature that was required has caused some spectacular failures, which we will discuss next time.

We will now consider the second main class of public key primitives in which encryption and signature are quite distinct.

2.2 Diffie Hellman

Recall Diffie Hellmann: Alice and Bob agree a prime p and a base g; Alice chooses a random number R_A, calculates $g^{R_A} \pmod{p}$ and sends it to Bob; likewise Bob chooses a random number R_B and sends $g^{R_B} \pmod{p}$ to Alice; they now both calculate $g^{R_A R_B} \pmod{p}$ and use it as a key.

The enormous power of this protocol is that it allows them to manufacture a shared secret 'out of whole cloth' as Diffie put it; the secrecy does not depend on previous shared secrets. This protocol is the main method used to provide 'forward secrecy': the property that even if my session key is compromised, I can still recover secrecy the next time I communicate. With care, one can implement even stronger properties; for example, one of the design goals of the STU-III mobile phone used by the US government is that even if all the secrets in a device are compromised by an enemy, it can recover confidentiality next time it interacts with an uncompromised device.

The main problem with the basic Diffie-Hellman protocol is that although Alice and Bob end up with a session key, neither of them has any idea who they share it with. Recall the 'manual' public key system in which a locked box is taken from Alice to Bob, who adds a lock and sends it back to Alice, who removes her lock and sends it back to him; the courier, Charlie, can break this protocol by playing Bob's role in an exchange with Alice and perhaps also Alice's role in a simultaneous exchange with Bob. In the same way, naive Diffie Hellman allows Charlie to set up simultaneous sessions with both of them, claiming to Alice that he is Bob and to Bob that he is Alice.

For this reason, there are a number of 'authenticated key exchange' protocols which combine Diffie-Hellman with some means of authenticating the exchange, which may use a shared password, signature keys or even real-time speech interaction. The precise security goals, and the design detail, require some care.

In 1b we mentioned one secure telephone product where the two principals would read out an eight digit hash of the key they had generated and check that they had the same value before starting to discuss classified matters. However, this was vulnerable to a small subgroup attack. The multiplicative group modulo p has order $p-1$, and so it has a subgroup of order 2. So the enemy could intercept Alice's $g^{R_A} \pmod p$ and replace it by $g^{R_A \cdot \frac{p-1}{2}} \pmod p$; similarly Bob's $g^{R_B} \pmod p$ is replaced by $g^{R_B \cdot \frac{p-1}{2}} \pmod p$ and the two principals end up using as a key $g^{R_A R_B \cdot \frac{p-1}{2}} \pmod p$, which has only two possible values. For this reason it is prudent not to choose the base g to be a primitive root, but a generator of a prime order subgroup modulo p. (If $p-1 = qw$ where q is a large prime, and γ is a primitive root modulo p, then set $g = \gamma^w$.)

2.3 ElGamal type schemes

Diffie and Hellman realised that their scheme could be used to give a public key cryptosystem with a simple twist: instead of exchanging g^{rx} in real time, this value is used as a public key and kept in a directory.

An innovation due to ElGamal makes the Diffie-Hellman idea even more powerful. This is to add an additional session key g^k.

Let Alice's private key be x, and her public key $y = g^x$. Then in order to encrypt a message M to send to her, the idea is to send BM, hint(B) where B is a blinding factor. A suitable blinding factor is y^k, where k is the randomly chosen session key. In detail, the encrypted message is

$$g^k, y^k M$$

Alice can decrypt this by raising the hint g^k to the power x (her secret key), getting y^k; she can then divide out the second term to recover M.

Encryption we could do already; the real power of session keys is that we can also construct digital signature schemes. Again, we will use a session key k and send a hint g^k. The signature will also be two numbers – conventionally r and s – where $r = g^k$ and s is given by a linear relation. For the ElGamal signature, we have

$$rx + sk = M$$

while for the Schnorr signature, the linear relation is

$$s = k + xm$$

The equations needed to verify these signatures are left as an exercise. However, note that many ElGamal type schemes are vulnerable to poor choices of public parameters (see exercise).

The US digital signature standard (or algorithm), which you may see referred to as DSA or DSS, is a variant on the ElGamal signature scheme in which the generator g is chosen to be a generator of a prime order subgroup of the multiplicative group of integers modulo p. In addition, the session key is transmitted as $(g^k \bmod p) \pmod q$, which blocks a number of attacks on ElGamal signatures based on the choice of degenerate parameters. DSS is used increasingly widely in military systems; for example, the Key Exchange Algorithm (KEA) used for US government and NATO applications combines a variant of Diffie Hellman with the Digital Signature Standard.

ElGamal type encryption and signature can also be performed in groups other than subgroups of the integers modulo n. For example, an *elliptic curve* is defined by the equation $y^2 = x^3 + ab + b$ and a group law can be defined on the points of such a curve based on the geometrical intuition that the identity (0) is the point at infinity and three points sum to zero if and only if they lie in a straight line. When elliptic curves are constructed over finite fields, it is believed that the discrete logarithm problem is in general hard. Until recently elliptic curve cryptosystems had the advantage of not being covered by certain patents; now that the Diffie Hellman patent has expired, they still offer some advantages over conventional ElGamal systems (smaller key sizes and faster computation). They have been proposed in a number of smartcard applications, but some people remain way of the greater conceptual complexity or are apprehensive that a clever mathematician will find a shortcut way of computing elliptic logarithms.

3 Threshold signature and decryption

We have seen that dual control is commonly used In business (in the Clark-Wilson model). It is also common for businesses to insist on multiple signatures on cheques; a simple rule might be 'any two directors can sign'. If we are going to use digital signatures on cheques in the future, we might want a system whereby a signature can only be generated by k out of n keyholders cooperating to sign the message.

Shamir and Blakley pointed out shortly after the invention of RSA that a k out of n scheme can be constructed by taking a polynomial of degree $k-1$ and letting the secret key d be its value at 0, so $p(x) = x^{k-1} + r_1 x^{k-2} + \ldots + r_{k-2}x + d$ where the r_i are random. We now give each of our company directors a point on the curve – called a 'share' of the key – and given any k of these, d can be worked out and signatures computed.

From the systems point of view, it would be preferable if the directors could use their shares to do their part of the signature computation independently on separate machines, so that no single security compromise would enable an attacker to collect enough material to forge signatures. But this is easy. Consider the Lagrange interpolation formula, which we use to compute a point on a curve of degree $k-1$ given k separate points (x_i, d_i) on it:

$$p(x) = \sum_{i=1}^{k} d_i \prod_{j=1, j\neq i}^{k} \frac{x - d_j}{d_i - d_j}$$

So $S = M^d = M^{p(0)}$ will have the form m to the power (sum of terms) where the only secret in each term is the key shadow of a single director; so M^d can be computed by the directors individually applying their shadows to create partial signatures, and these partial signatures can then be multiplied together to get the real signature.

This threshold signature scheme can also be used for decryption. It is used in products (AT&T's Rampart and IBM's Proactive Security) which seek to distribute trust; one can, for example, replace a secure server with several less secure ones and get them to vote on transactions.

4 Digital Cash

An example of the power of public key mechanisms is given by 'digital cash'. The idea here is to create 'digital coins' whose authenticity anyone can verify, and which still enable users to spend money anonymously without the bank being able to trace their purchases.

The basic idea is that a coin is a string of a certain format, signed with a certain RSA key. It might for example be $(m, h(m))$ where m is a random serial number and h is a collision-free hash function. If the customer makes up such a string c, and the bank's RSA public key is N, e, then the customer will think of a random number R and present the bank with $R^e c$. The bank will generate a signature on this – $(R^e c)^d \pmod N$ – having debited the customer the value of the coin. This value is just $R.(c^d) \pmod N$ and the customer can now divide by R to get a valid coin whose serial number is unknown to the bank.

An actual implementation requires a little more complexity because of the risk that the customer might spend the same coin in multiple shops (even if the coins are kept in smartcards, they might always be probed). The usual solution is to encode the customer's identity in the coin in a way that is normally protected, but is revealed if he spends it twice (the trick is to have a challenge-response protocol between the customer and the merchant when the coin is spent; if the same coin is spent twice, then different challenges have to be answered and this gives away the customer's identity).

Of course, much more is needed for anonymity that just the cryptographic payment mechanisms; one needs anonymous communications channels as well.

5 Further Reading

For further information on public key cryptography, see Schneier's *'Applied Cryptography'*. The classic papers are also well worth a read. They are 'New Directions in cryptography' by Whitfield Diffe and Martin Hellman, in *IEEE Transactions in Information Theory vol IT-22 no 6 (November 1976) pp 644–654*, and "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", by Ron Rivest, Adi Shamir and Len Adleman, in *Communications of the ACM v 21 no 2 (February 1978) pp 120–126.*

As for the underlying number theory, the average college library will have a lot of relevant books for the mathematical Tripos, and some of them are fairly readable. The classic text is Hardy and Wright's *'The Theory of Numbers'*: this includes all the proofs which we have omitted (or sketched) in these notes. Another good book, and perhaps easier for the computer science student, is *'A Course in Number Theory and Cryptography'* by Koblitz (Springer Graduate Texts in Mathematics no 114). This goes into detail on how the number theory is applied in a range of cryptographic algorithms.

6 Problems

1. Prove that there are infinitely many prime numbers of the form $6n + 5$, and also of the form $4n + 3$.
2. Show that if the generator $g = 2$ is chosen, then ElGamal signatures can be forged on any message.
3. Write out the full formulae for threshold decryption and signature.
4. The US digital signature standard assumes a prime p of 512-1024 bits, a prime q of 160 bits with $q \mid (p-1)$, an element g of order q in the integers modulo p, a secret signing key x and a public verification key $y = g^x$. The signature on a message M is (r, s) where

$$r = (g^k \pmod p) \pmod q$$

$$s = (h(M) - xr)/k \pmod q$$

Write down the equation(s) used to verify such a signature.

The standard also specifies that r must lie strictly between 0 and q. What might go wrong if an application developer failed to check this?

5. Find an explicit formula for the secret exponent d in the RSA cryptosystem given a public exponent of $e = 3$. Can you generalise this to arbitrary e? Is your formula a faster way of finding d than Euclid's algorithm?

6. Show that if the same message is encrypted and sent to multiple recipients using RSA with $e = 3$, then the message can be deciphered without knowing the factors of any of the keys.

7. Show that if we do RSA with exponent 2, breaking the algorithm is strictly equivalent to factoring. How would decryption be done in this case?

8. Some systems do Diffie-Hellman with a composite modulus. Prove that someone who can compute discrete logs modulo the product of two primes can factor this product. Then extend your result to arbitrary composite numbers.

9. A protocol called Oakley has been recommended in an IETF draft for key establishment in IP v 6. Find the documentation on the web, and extract a concise description of it (using appropriate notation). is it sound?

10. Perform the same exercise for SSL, SSH, SET and PGP.