PGP key validity: is keyowner id correct? ownertrust: is key owner reliable?

key validity is computed from trust in signers, only considering signers with key validity complete (or better)

complete is assigned when key is signed by at least one user with ownertrust complete or by at least x users with ownertrust marginal

marginal is assigned when key is signed, but by less than x users with ownertrust marginal

unknown is assigned when key is signed by no one with ownertrust at least marginal

ownertrust is computed from owner trust of signers, only considering ultimate valid keys

certpath validation: shell model: all signatures (leading up to root) must be valid at verification time

modified/hybrid model: all signatures must have been valid at the point in time where the signature closest to the verifier was made

chain model: all signatures must have been valid at the point where they were made

trust models (hierarchy): trusted list: every participant has list of CAs. common root: one root signs all CAs

cross-certification: root (or intermediate) CAs sign each other's certs. bridge CA: a bridge CA is trusted by CAs to further delegate trust to other parties

hybrid encryption: the use of asymmetric crypto to securely exchange a symmetric key

public key certs bind public keys to subjects

direct trust is the verification/exchange of keys directly with their owner. it scales poorly, but can be used to obtain initial trust anchors

certificate policy (CP) states what policy should be complied with. certificate practice statement (CPS) states how to comply

if a cert contains both key usage and extended key usage extensions, the cert must only be used for purposes consistent with both of them

practical problems include: key availability, key ownership, key validity, performance

public key encryption — relevant for signatures

a CRL is a signed list of revoked certificates (by serial number). A CRL contains the time at which it was produced & the time of next update. They tend to grow quite large. CRL types:

over-issued CRL: issued more often than only at next update time. allows better load distribution

delta CRL: contains changes since last base CRL. provides smaller CRLs

indirect CRL: CRL issuer ≠ cert issuer. allows different security levels for CRL/cert signing

segmented CRL: separates revocation information into multiple CRLs

redirect CRL: CRL which points to actual CRL; easy distribution

OCSP allows clients to request status of one or more certificates by querying a responder. Responses are signed, and can be: unknown (nothing is known about the certificate), revoked or good (certificate not revoked, but may have expired or may not exist). Signed responses can be stored and provided/stapled as proof of validity in nearby future.

novomodo includes $H(R)$ and $H^{365}(T)$ in cert. On day i, R is published if cert revoked, and $H^{i}(T)$ if cert still valid. Cert only needs to contain 2 hash values, while status info consists of 1 hash value.

In certificate transparency, certificates are only valid when they are included in a blockchain-like public log. This aims to make it impossible for CAs to secretly issue valid certificates.

IPsec | AH: authentication header | ESP: encapsulating security payload | transport mode is used between hosts, the header is not encrypted (in ESP), but parts of it are authenticated (in AH); only the payload is protected (from destination to destination)

in tunnel mode, the entire IP packet is protected (including header) and becomes payload of new IP packet. tunnel mode can be used between hosts, gateways, or between host and gateway.

a key encapsulation mechanism (or KEM) encapsulates the symmetric key used for hybrid encryption using asymmetric cryptography

Wireguard is a simple VPN protocol over UDP, based on the noise protocol framework with IK

TLS provides transparency for higher layers as it runs on top of TCP. A TLS record can have different types/protocols:

handshake: initiate session, which includes authenticating server & client and establishing keys

application: data transfer, which includes computing MACs for integrity and encrypting MACs and data

alert: alert the other side of exceptional conditions (errors and warnings)

the pre-master secret is the value obtained from key exchange, which is then used to construct the master secret.

when using DH(E) or ECDH(E) the pre-master secret can be described as the result of a Diffie-Hellman key exchange

when using RSA, the pre-master secret is a randomly generated bit-string encrypted by the client for the server

noise protocol framework:

e = ephemeral (key)
s = static
ee, se, es, ss = DH results where first letter indicates initiator key type and 2nd letter indicates responder key type

first character in (A)KE case:
N = no static key for initiator
K = static key for initiator, known to responder
X = static key for initiator, transmitted to responder
I = static key for initiator, immediately transmitted to responder

second character in AKE case:
N = no static key for responder
K = static key for responder, known to initiator
X = static key for responder, transmitted to initiator

Wireguard achieves KCI security for authenticity, has perfect forward secrecy, avoids replay attacks and has identity hiding

h = handshake hash (hash of all handshake messages)

CK = chaining key (hashes all previous DH outputs; used to derive encryption key)

payload encryption: whenever new CK is set, derive session key k and new nonce n; whenever k is set, start encrypting payload and use h as associated data

SSL strip: intercept HTTP traffic & replace links/redirects to HTTPS with HTTP

BEAST (Browser Exploit Against SSL/TLS): in CBC mode, we can check whether $p_0 = x$ by doing $p_2 = x \oplus c_0 \oplus c_1$. Requires attacker to be able to make client send data (i.e. the chosen guess p_2)

CRIME (Compression Ratio Info-leak Made Easy): second occurrence of a character is encoded as a back reference in compressed plaintext, which leads to shorter ciphertext; allows for guessing secrets in ciphertext

BREACH (Browser Reconnaissance and Exfiltration via Adaptive Compression of Hypertext): same as CRIME, but applied to HTTP compression instead of TLS encryption

padding oracle: error message depends on correctness of padding in plaintext; allows for checking validity of plaintext by guessing based on crafted message.

Lucky 13: padding oracle which uses timing of MAC computation as a side channel

POODLE (Padding Oracle On Downgraded Legacy Encryption): make client accept lower TLS version, then apply padding oracle

FREAK (Factoring RSA Export Keys): use MitM to downgrade key systems used for symmetric key exchange to export-grade crypto, then factor weak RSA key

Logjam: similar to FREAK, but for Diffie-Hellman with standard primes

Heartbleed: buffer overflow in heartbeat message implementation of OpenSSL

Sweet32: attack on 3DES-CBC; aim is to search for a colliding ciphertext with a known plaintext (or: a birthday attack on 64-bit block ciphers)

DROWN (Decrypting RSA with Obsolete and Weakened eNcryption): use of SSLv2 as a Bleichenbacher oracle to decrypt a TLS handshake.

RC4 is a stream cipher which suffers from many biases and correlations.

Fluhrer, Mantin and Shamir attack: if nonce and long-term key are concatenated (as in WEP), long-term key can be discovered (key of RC4)

in messaging, deniability may be a desired security property (on top of CIA). Deniability can be achieved using MACs; if a message has a valid MAC, it must have come from the communication partner. However, due to both parties being able to create a MAC, this proof is not transferable.

OTR protocol: initial key share authenticated with long-term private keys; later key shares auth'd by MAC; HMAC used for all encrypted messages; forward secrecy guaranteed because long-term keys only used for signing encryption keys (not for encryption itself); after a round trip, MAC keys are 'leaked'/published in next iteration, which provides deniability; both users must be online

multi-party OTR is roughly pairwise OTR in groups; it does not have perfect forward secrecy during the communication phase → this is used to authenticate a group key which is used for the actual group chat

socialist millionaire problem: check whether wealth (or shared secret) is equal without revealing how much one owns. (non-socialist) millionaires protocol checks whose wealth is greatest

Silent Circle Instant Message Protocol (SCIMP): leaking of the encryption key leaks all following messages in the session; authentication is done out of band; both parties have authentication/encryption key, providing deniability (since either one could have written the message)

Signal protocol: combines DHE forward-secure key update with hash-based forward-secure key update; authenticates keys by mixing in previous authenticated keys. A new chain of keys is derived using the combination of: 1. the private key whose public key was last sent to the other party 2. the public key which was last received from the other party.) Hence, as soon as one changes from receive to send, newly sent messages cannot be decrypted anymore with earlier/existing keys. Similarly, once one changes from send to receive, newly received messages cannot be decrypted anymore with earlier/existing keys. Forward secrecy means that even after a key compromise, previously ... [illegible]

OTR v2 and Signal can recover from a temporary key compromise

when passwords are hashed using the same function, the same hash is returned for the same password. This allows for pre-computing hashes to save time. workaround: use a salt, and store the hash of the salt concatenated with the password.

In a hash chain, a hash function is applied multiple times in a row (with a function in between to reduce output to domain of hash function)

PBKDF2 (password-based key derivation function 2) uses a pseudorandom function (e.g. HMAC-SHA) a password and a salt to derive a key. downsides: depends on hash function designed for speed, uses little memory

Bcrypt uses Blowfish encryption, a cost, salt and password to derive a key. The cost factor accounts for changes in computing power, but the function requires little memory.

scrypt has one cost factor which is used for both time and memory. It is based on PBKDF2 and allows for an attack based on a time-memory trade-off, since the memory requirements are based on back references to earlier outputs in a hash chain. note: scrypt is relatively new

Time-space trade-offs in password hashing enable the use of GPUs/ASICs (for attacks)

argon2 is a function which first expands to the entire available memory, then applies a sequence of memory-hard transformations, and concludes by absorbing the entire state into a small tag. It allows separate parameters for time and memory cost.

online attacks attempt to login, but can be defeated by means of rate-limiting; offline attacks check for password correctness

diceware is a means of constructing passwords from several random words; it remains easy to enumerate using dictionary

a risk of using security questions to restore passwords is that the answers may be easier to guess than the password itself anonymity: TCP/IP and TLS → no anonymity; IPSec → anonymity only in tunnel mode, and only for gateways (between gateways)

dining cryptographers problem: 3 cryptographers on dinner, someone paid, want to figure out if it was one of them without revealing who it was. 1. share pairwise secret bits (between AB, BC, AC) 2. everyone announces a message the XOR of their shared bits 3. if someone paid, they XOR their message with 1 4. broadcast all messages 5. XOR over all messages; result = 0 if no one paid, result = 1 if someone paid

cryptographic mixing: to send message m from A to B, m is sent to the mixer in the format (note: M is a mixer) $E_{K_M}(B, E_{K_B}(m))$. M forwards the message to B, who can decrypt it

flushing modes: message threshold: wait until n messages are received, then release all. message pool: pool size n, probability p. After n messages in pool, shuffle, then send each with probability p. Unsent messages remain in pool. stop and go: under determined waiting time for each message

adversary properties: internal/external: can compromise communication medium (external) and mix nodes/recipients/senders (internal). passive/active: active can arbitrarily modify computations & messages (insert/delete); passive can only listen. static/adaptive: static chooses compromised resources up front, adaptive can change resources under control during protocol execution

Countermeasures against mixnet attacks include padding and dummy traffic

In re-encryption mixnets the ciphertext is re-randomized at each node instead of being decrypted. This uses a re-randomizable form of ElGamal encryption. advantages [illegible]
disadvantages: 1. cannot use hybrid encryption 2. routing done by input [illegible] (but possible with [illegible])

electronic cash: conventional cash properties: difficult to forge, transferable, anonymous, untraceable, unforgeable, hard to duplicate. simple scheme → needed protocols: deposit, payment, withdrawal. initial solution to double spending problem: give digitally signed bill to bank, only accepts if not deposited before

cut and choose: 1. create k checks with random serial numbers and amount 2. send envelopes (with carbon paper, so that signature is done on check) to bank and tell the amount 3. bank opens k−1 envelopes at random and verifies amount 4. bank signs remaining envelope, withdraws amount and sends back envelope 5. user takes out check and verifies signature

blind signatures: blind signatures allow to hide the message that is being signed. That is, the signer is provided with the blinded message b(M), then provides a signature on σ' on b(M), from which a signature σ on M can be extracted. security properties: one-more unforgeability: from l oracle calls, no more than l valid signatures can be obtained; unlinkability: the signer should be unable to link any particular signature with a specific execution of the signing protocol (this is possible using RSA signatures)

multiple exponents solution: use different public exponents (e_i) to encode different denominations of electronic coin. As long as the e_i are relatively prime, index-calculus-like attacks do not work

offline version of the protocol: include $y_i = H(x_i)$ and $y_i' = H(x_i')$ in the bill, where the XOR of x_i and x_i' gives the user's identity. Then use the approach with unblinding k−1 bills again (contents of this). If the user attempts to spend a bill twice, (a part of) their identity is leaked

Bitcoin: transactions contain many inputs, maximum two outputs. A block includes a periodical ('timestamped') snapshot of all transactions. Proof of work is a de facto distributed timestamp [illegible]. hard for block, the target is adjusted to always take 10 minutes. Rewards [illegible]
malicious behaviour [illegible]

homomorphism is a structure-preserving map. ElGamal is a homomorphic encryption scheme, in which multiplying two ciphertexts is equivalent to multiplying the corresponding plaintexts. In exponential ElGamal, the (two) ciphertexts are first used as exponents, i.e. m_0 and m_1 become g^{m_0} and g^{m_1}. This way, when the ciphertexts are multiplied, in effect the corresponding plaintexts are added. This requires solving a discrete log to retrieve the plaintext, however, which is doable as long as the numbers/space to search is small.

IND-CCA2 (indistinguishability under adaptive chosen ciphertext attack) security is unachievable for any homomorphic encryption scheme. IND-CCA1 is achieved by ElGamal, in addition to IND-CPA. Possible homomorphic encryption schemes are rerandomizable

CGS voting scheme (basic version): basic version involves a trusted authority to which all votes are encrypted and which can learn all votes. The resulting ciphertext contains the sum of all votes, which (due to homomorphic properties) can be decrypted to the actual sum of votes. The basic version also assumes that the votes are boolean (i.e. for/against). The role of a trusted authority can be distributed by encrypting to the product of keys of several parties; only the group of parties having the key can decrypt. This does allow for obstruction by a party unwilling to collaborate, which can be overcome with threshold schemes.

Fully homomorphic encryption (FHE) allows cloud computing without sharing data. Bootstrapping is a process using the encryption of a secret key with its public key. This allows for 'decrypting' without knowing the key, and is used (by encrypting and decrypting) to reduce noise in lattice based computation schemes.

properties of zero-knowledge proofs: 1. correctness → honestly generated proofs of true statements are accepted 2. soundness → it is impossible to create valid proof for a false statement 3. zero-knowledge → it can be simulated for false statements such that the result is indistinguishable from a real proof. The Fiat-Shamir heuristic allows for turning interactive ZK proofs into non-interactive ZK proofs (which can be seen as signatures), under the assumption that random oracles exist.

Commitment schemes require two properties: hiding (of the content of the commitment) and binding (so that the content of the commitment cannot be changed). These are deterministic, hence leak information and don't hide. A random oracle suffices if the message is padded by a sufficiently long random string.

Pedersen commitments are a scheme with common reference string $h = g^x$. To commit on exponent m, one samples random r and publishes the commitment $g^m h^r = g^{m+xr}$. Under the assumption that x is unknown this binds. It also achieves information-theoretic hiding.

multi-party computation: GMW scheme → a scheme which reduces multi-party computation of AND to two-party computation. The scheme [illegible] both AND and XOR, which form a complete set of operations.

Oblivious transfer is a method for B to send one message out of two to A without A having to tell B which message she desires. A mechanism exists which is only secure with passive adversaries. In this mechanism, A sends two public keys to B, where she has the private key for only one of the private keys. This does not work against active adversaries (although such a scheme is possible).

security properties of MPC: 1. completeness: if all parties are honest, the end-results are correct. 2. fairness: either all parties or no party receives their result. 3. soundness: the scheme is secure. Fairness requires an honest majority, an arbiter scheme, or encrypting results and sharing the keys one bit at a time.

simulation-based security notions: The adversary may corrupt a subset of all participants, and learns their in-/outputs and some pre-defined leakage (e.g. message length). The scheme is secure if a simulated transcript (under idealized functionality) of in-/outputs is indistinguishable from a real transcript.

It is possible for a scheme to be secure against active attacks while being insecure against passive attackers.

one-time signatures: message signed bit-by-bit. Lamport's 1-time signature scheme: sign by releasing part of a secret, the preimage of the public key (i.e. public key = H(secret)). Each signature has 2·256 hash outputs (32 bytes each) as public key (256×32 bytes). Winternitz 1-time signature: public key = H^{16}(secret). Reveals H^m(SK) as signature, which can be checked using pk = H^{16-m}(s). To make a secure system, use two hash chains in opposite directions. Merkle's multi-use signature scheme → merges one-time public keys into a single public key by combining them through a Merkle tree; the public key is now the hash of the combined keys. Only the top node needs to be released as the public key for this to work.

attack goals: full break (FB) A can compute secret key; universal forgery (UU) A can forge signatures for any message; selective forgery (SU) A can forge signature for a message chosen before start of the attack; existential forgery (EU) A can forge signature for arbitrary message for which signature was not obtained from oracle during attack

attack models: key only attack (KOA) A only gets the public key; random message attack (RMA) A gets public key & signatures on a set of random messages; adaptively chosen message attack (CMA) A learns public key and is allowed to adaptively (i.e. based on/after obtaining target) ask for signatures on messages of its choice

attacks on RSA signature schemes: 1. in textbook RSA, encrypting a random string σ produces a message M for which σ is a valid signature 2. due to homomorphic properties, the product of two textbook RSA signatures is a valid signature on the product of the messages. 3. in a blinding attack, the message to be signed is hidden by multiplying it by r^e for some random r. This r contributes $r^{ed} = r$ mod n to the signature obtained from the oracle on $r^e m$ and can be divided out. 4. a scheme in which the hash of a message is used is vulnerable to index calculus attacks, especially for short hashes

In the standard model, it is assumed that a building block has a property P, which is then used in a reduction. In an idealized model, it is assumed that a building block behaves perfectly; the building block is replaced by an oracle in a reduction. The random oracle model replaces hash functions with a random oracle, which always returns a uniform random (but consistent) result. It can be implemented in theory by lazy sampling: return result if stored, and return uniform random sampled value (which is then stored) if the result is not yet stored. The random oracle model is a heuristic model.

RSA-FDH (full domain hash) is a signature scheme where the hash function maps to the entire domain of the RSA group (i.e. $\mathbb{Z}$ mod n). This provides a sufficiently large space of possible hashes to prevent index calculus attacks

Block cipher modes: Electronic Code Book (ECB) permutes all blocks in the same way. Does not remove structure/patterns in data. Counter (CTR): the permutation applied to a block depends on its number. This is vulnerable to manipulation of the contained message (since blocks are treated independent of each other). Tweakable block cipher: accepts an additional 'tweak' and is indistinguishable from a random permutation per tweak and per key. Offset Codebook method 2 (OCB2): block cipher mode which had a security proof and was broken anyway. The proof placed unrealistic restrictions on the attacker

AE(AD) stands for Authenticated Encryption (with Associated Data). It implies the following: 1. Ciphertext does not reveal information about plaintext (except length) 2. it is impossible to create a valid ciphertext without knowledge of the key. Both properties hold even when the adversary has access to encryption/decryption oracles and is able to choose nonces.

OCB2 forgery: a forgery for OCB2 can be constructed from an encryption oracle query for (N, A, M), with $M = len(0^n) \| m_2$, where $|m_2| = n$, A = ε and N is arbitrary. Some modifications allow for extending this attack to longer messages. Some more work allows for constructing universal forgeries and recovering plaintexts

post-quantum cryptography: Grover's algorithm reduces time for brute force by square root. Introducing new cryptographic systems and making them widely used takes very long (RSA to ECC switch took 30 years). Shor's algorithm for period finding breaks RSA, ECC, DH.

categories of public-key post-quantum systems: hash-based signatures based on hardness of finding second preimages to hash functions; code-based encryption based on hardness of decoding error-correcting codes; isogeny-based encryption: based on hardness of finding isogenies between elliptic curves over finite fields; lattice-based encryption & signatures: relies on hardness of finding short vectors in some (typically special) lattice; multivariate-quadratic signatures: relies on hardness of solving systems of multivariate equations over finite fields

schemes from NIST competition

name	function	hardness assumption	
Kyber	KEM	structured lattices	winners
Dilithium	signature scheme	structured lattices	winners
Falcon	signature scheme	structured lattices	winners
SPHINCS+	signature scheme	hash functions	winners
BIKE	KEM	codes	advanced to round 4
Classic McEliece	KEM	codes	advanced to round 4
HQC	KEM	codes	advanced to round 4
SIKE	KEM	isogenies	advanced to round 4, now broken

requires storing 2^n one-time secret keys, but this can be optimized by generating them from a (short) seed. The scheme could be made stateless by using it to sign other trees to expand the key space (this idea was due to Goldreich & Levin).

Winternitz equations: each chain runs for 2^w steps. For signing 256-bit hash, this needs $t_1 = \lceil \frac{256}{w} \rceil$ (ceiling) chains. m is written in base 2^w as $m = (m_{t_1-1}, \ldots, m_1, m_0)$ with zero-padding if needed. Then put $C = \sum_{i=0}^{t_1-1} (2^w - m_i)$ where $C \le t_1 2^w$; write C in base 2^w which takes $t_2 = \left\lceil \frac{\lfloor \log_2 t_1 \rfloor + 1}{w} \right\rceil$ (floor; plus one) w-bit integers $C = (c_{t_2-1}, \ldots, c_1, c_0)$. publish $t_1 + t_2$ public keys, sign with chains of lengths $m_{t_1-1}, \ldots, m_1, m_0, c_{t_2-1}, \ldots, c_1, c_0$