Distributed Systems

Rik Sarkar
James Cheney

Security Protocols & Case Studies
March 17, 2014

Recap & outline

- Security is hard
- Cryptography is not security
- No "security through obscurity"
- Today:
  - cryptographic protocols, continued
  - case studies: Kerberos, TLS, Wifi, Bitcoin

Scenario 3. Authenticated Communication with Public Keys

- Alice and Bob want to securely set up a shared private key K_{AB}
- Assume that Bob has generated his own public/private key pair K_{Bpub},K_{Bpriv}
- We also assume that there is some **public-key certificate system** such that Alice can obtain Bob's public key in a way that she is confident that it is indeed Bob's public key
- Alice obtains Bob's public key K_{Bpub}
- Alice creates a new shared key K_{AB} and encrypts it using K_{Bpub} using a public-key algorithm. This she sends to Bob $\{K_{AB}\}K_{Bpub}$
- Bob decrypts this using the appropriate private key to obtain the shared private key K_{AB}.
- Alice and Bob can now use K_{AB} to communicate

Scenario 3. Authenticated Communication with Public Keys

- This is a hybrid cryptographic protocol and is widely used as it exploits useful features of both public-key and secret-key encryption algorithms
- The slower public-key algorithm is used to set up the speedier secret-key communication
- Problems:
  - The distribution of public keys. Mallory may intercept Alice's initial request to obtain Bob's public key and simply send Alice their own public key.
  - Mallory then intercepts the sending of the shared key which they copy and then re-encrypt using Bob's real public key and forward it to Bob.
  - Mallory can then intercept all subsequent messages since they have the shared secret key. They may also need to forward the messages on to Bob and Alice depending on the delivery mechanism.

Digital Signatures

- Cryptography can be used to implement **digital signatures**
- Alice can also encrypt the message using her own secret key
- Anyone can decrypt the message so long as they know Alice's public key
- Provided we can be sure that the public key in question really is Alice's we now know that the message must have originated from Alice, since only Alice knows Alice's private key
- Rather than encrypt the entire message Alice can compute a digest of the message, where a digest is similar to a checksum except that two distinct messages are very unlikely to have the same digest value
  - Cryptographically secure hashing: MD5 (broken), SHA-1
- This digest is encrypted and attached to the message, the receiver can then check that the unencrypted digest matches the (receiver computed) digest of the contents of the message

Scenario 4. Digital Signatures

- Alice wishes to sign a document M so that any subsequent receiver can be sure that it originated from Alice
- Alice computes a fixed length digest of the document Digest(M)
- Alice encrypts the digest with her private key and attaches the result to the message: M,{Digest(M)}K_{Apriv}
- Alice makes the document with signature available
- Bob (or anyone else) obtains the signed document, extracts M and computes d = Digest(M)
- Bob decrypts {Digest(M)}K_{Apriv} using K_{Apub} and compares the result to d, if they match the signature is valid.

Scenario 4. Digital Signatures

- We have three requirements of digital signatures
  - **Authenticity**: It convinces the recipient that the signer deliberately signed the document and it has not been altered by anyone else
  - **Unforgeability**: It provides proof that the signer, **and no one else**, deliberately signed the document. In particular the signature cannot be copied and placed on another document
  - **Non-repudiation**: The signer cannot credibly deny that the document was signed by them
- Note that encryption of the entire document, or its digest, gives good evidence for the signature as unforgeable
- Non-repudiation is the most difficult to achieve for digital signatures. A signer may simply deliberate disclose their secret key to others and then claim that anyone could have signed it.
- This can be solved through engineering but is generally solved through social contract "If you give away your secret key you are liable"

Scenario 5. Certificates

- Suppose Alice would like to shop with Carol
- Carol would like to be sure that Alice has some form of bank account
- Alice has a bank account at Bob's bank
- Bob's bank provides Alice with a certificate stating that Alice does indeed have an account with Bob.
- Such a certificate is digitally signed with Bob's private key K_{Bpriv} and can be checked using Bob's public key K_{Bpub}

Scenario 5. Certificates

- Now suppose Mallory wished to carry out an attack to convince Carol that she owns Alice's account (i.e. identity theft)
- This is quite simple, Mallory only requires to generate a new public-private key pair $K_{BprivFake}, K_{BpubFake}$
- She then creates a certificate falsely claiming that she is the owner of Alice's account and signs it using $K_{BprivFake}$
- This attack works if she can convince Carol that $K_{BpubFake}$ is the true public key of Bob’s bank

Scenario 5. Certificates

- The solution is for Carol to require a certificate from a trusted fourth party, Dave from the Bankers' Federation, whose role it is to certify the public keys of banks
- Dave issues a public-key certificate for Bob's public key K_{Bpub}. This is signed using Dave's private key K_{Dpriv} and can be verified using Dave's public key K_{Dpub}
- Of course now we have a recursive problem, since now we need to authenticate that K_{Dpub} is the legitimate public key of Dave from the Bankers' federation.
- We break the recursion by insisting that at some point Carol must trust one person, say Dave, and to do so may require to meet them in person to exchange public keys
- Note that Carol only has to trust Dave in order to verify bank account certificates from a variety of banks

Scenario 5. Certificates

- To make certificates useful, we require:
  1. A standard format such that certificate issuers and users can construct and interpret them successfully.
  2. Agreement on the way in which chains of certificates are constructed and in particular the notion of a trusted authority
- In addition, we may wish to revoke a certificate, for example if someone closes their account
- This is problematic since once the certificate is given it can be copied and stored etc
- The usual solution is for the certificate to have an expiration date
  - the holder of the certificate must periodically renew it (in the same way that one renews a passport)

Announcements

- **No lecture March 20 (Thursday)**
- **Final lecture March 24**
  - **review of course and examinable material**

Case studies

- We will now see how some of these ideas play out in practice:
  - Needham-Schroeder protocol and Kerberos
  - TLS (used in HTTPS)
  - IEEE 802.11 WiFi security
  - Electronic money - Bitcoin

Needham-Schroeder

- Authentication in more detail:

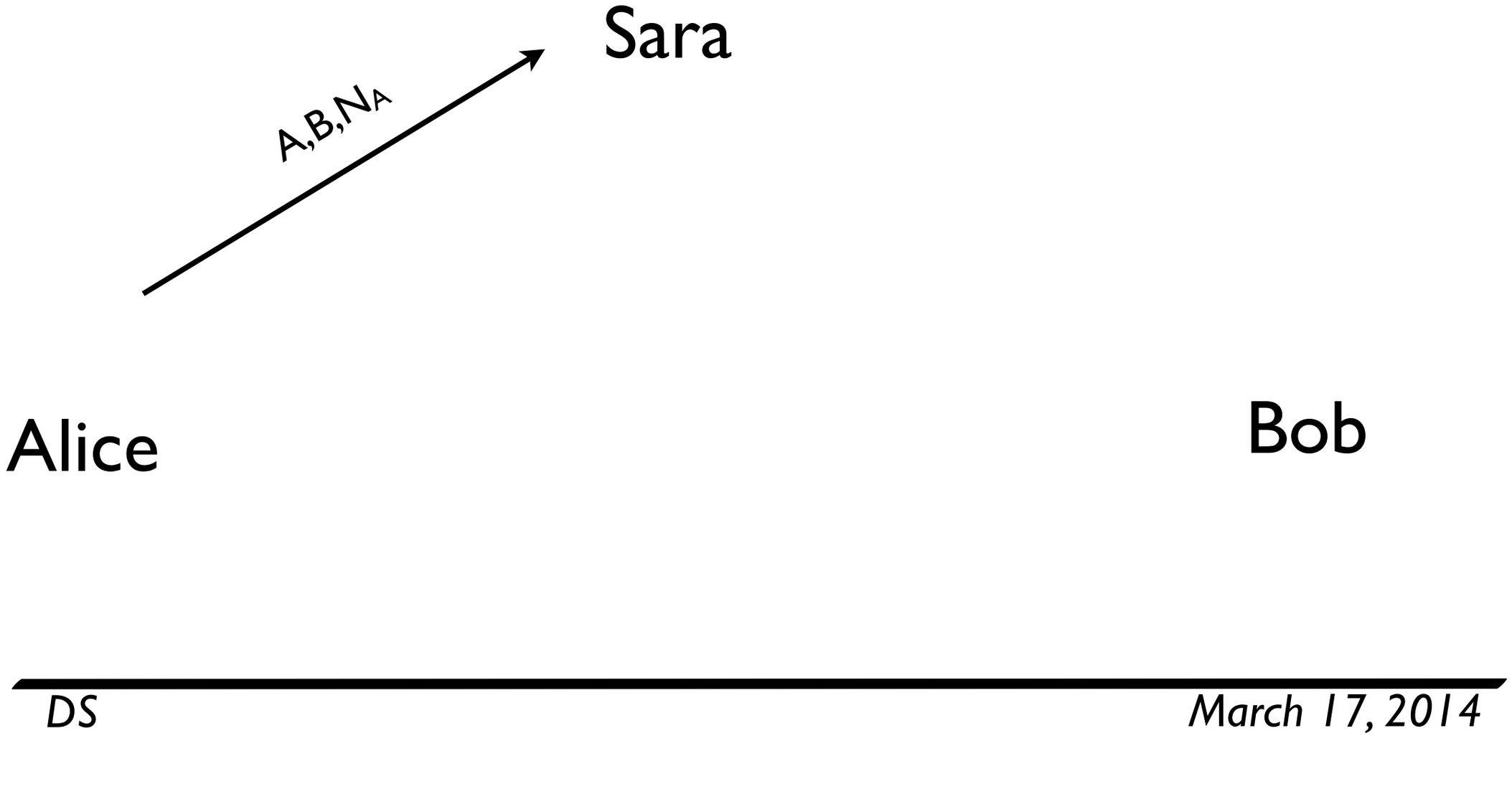
Sara

Alice

Bob

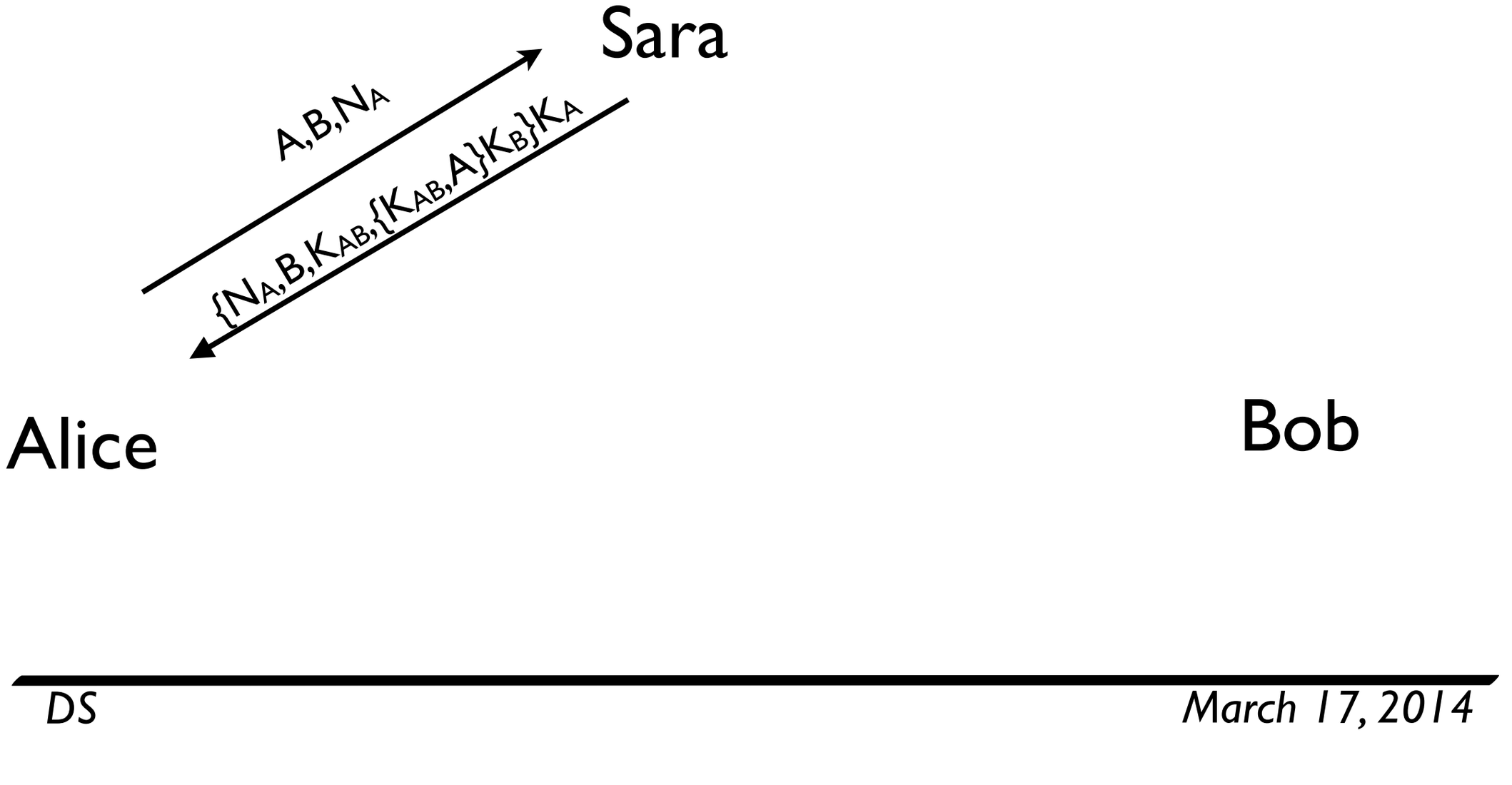
Needham-Schroeder

- Authentication in more detail:



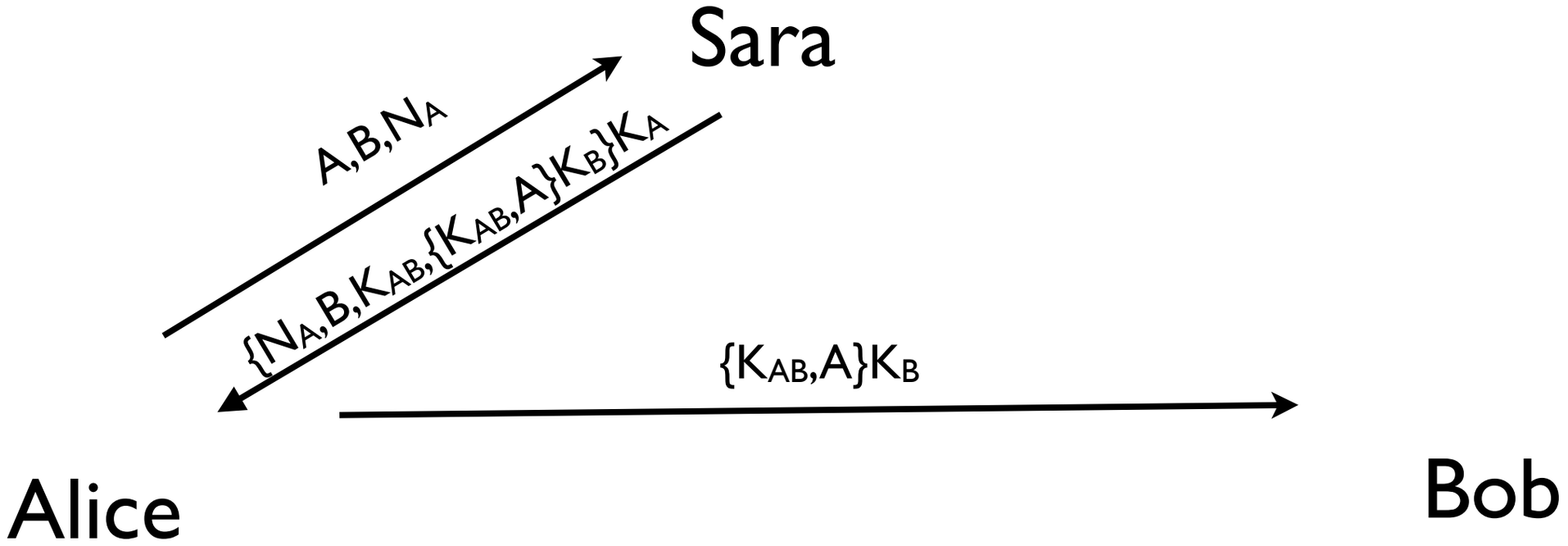
Needham-Schroeder

- Authentication in more detail:



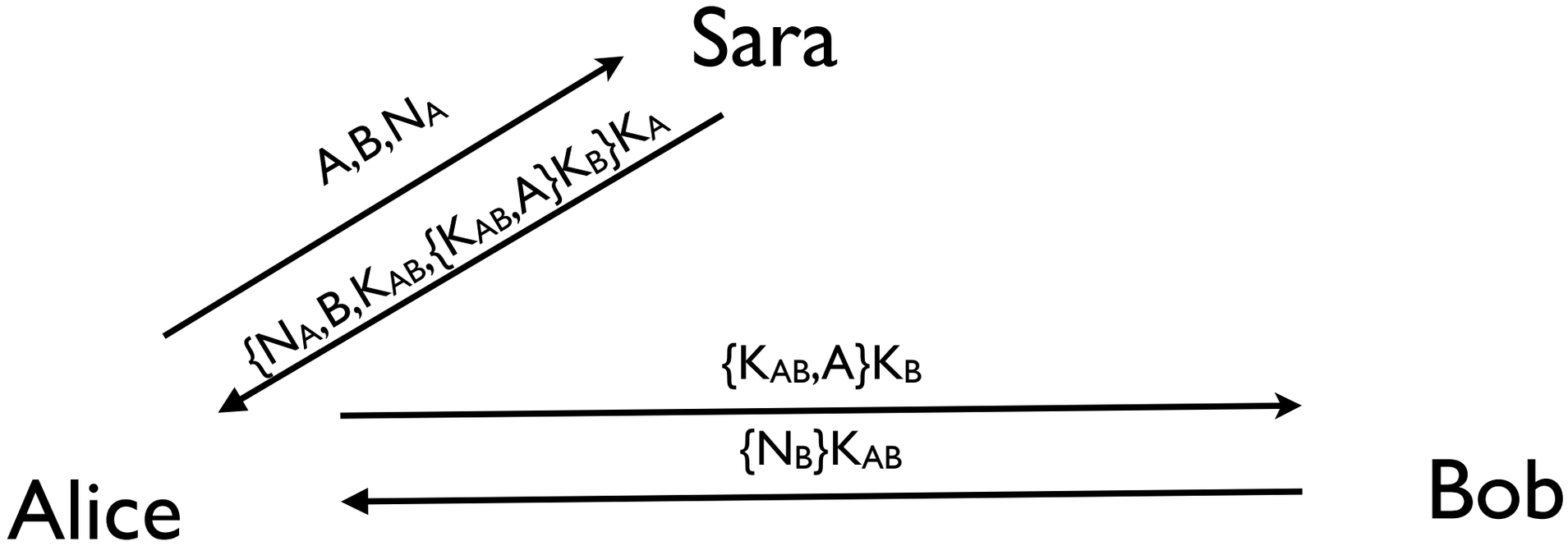
Needham-Schroeder

- Authentication in more detail:



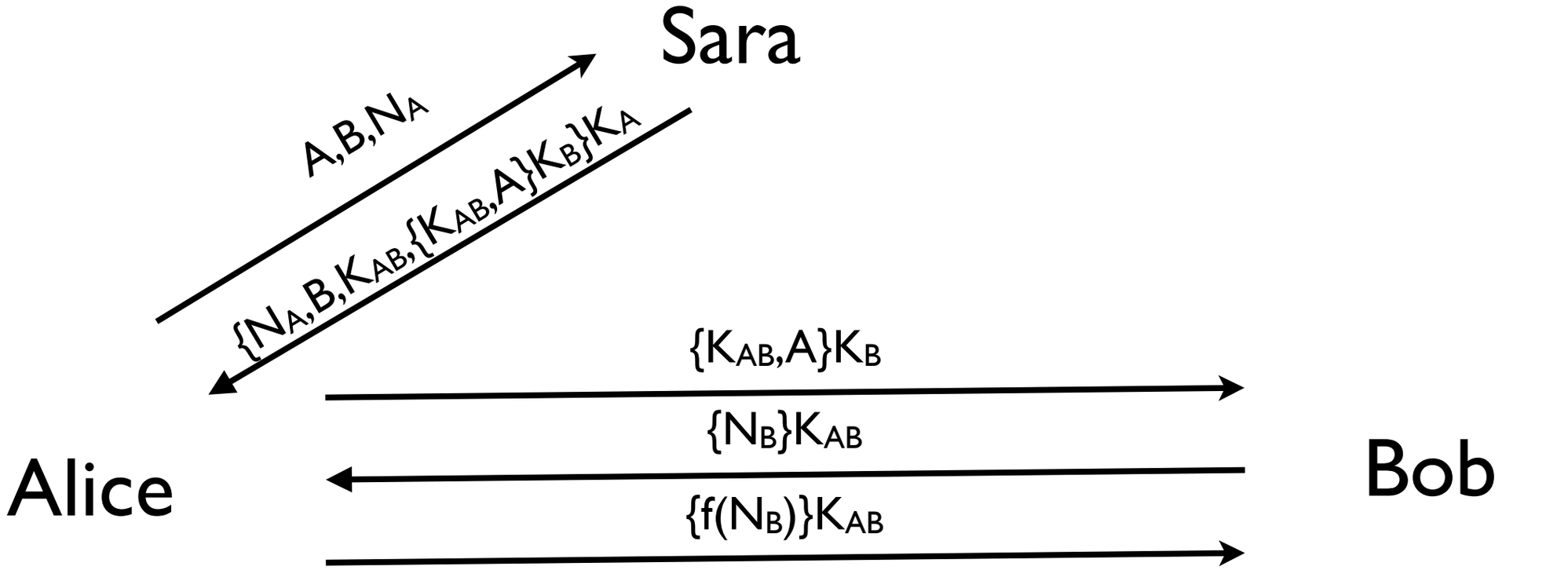
Needham-Schroeder

- Authentication in more detail:



Needham-Schroeder

- Authentication in more detail:



Nonces

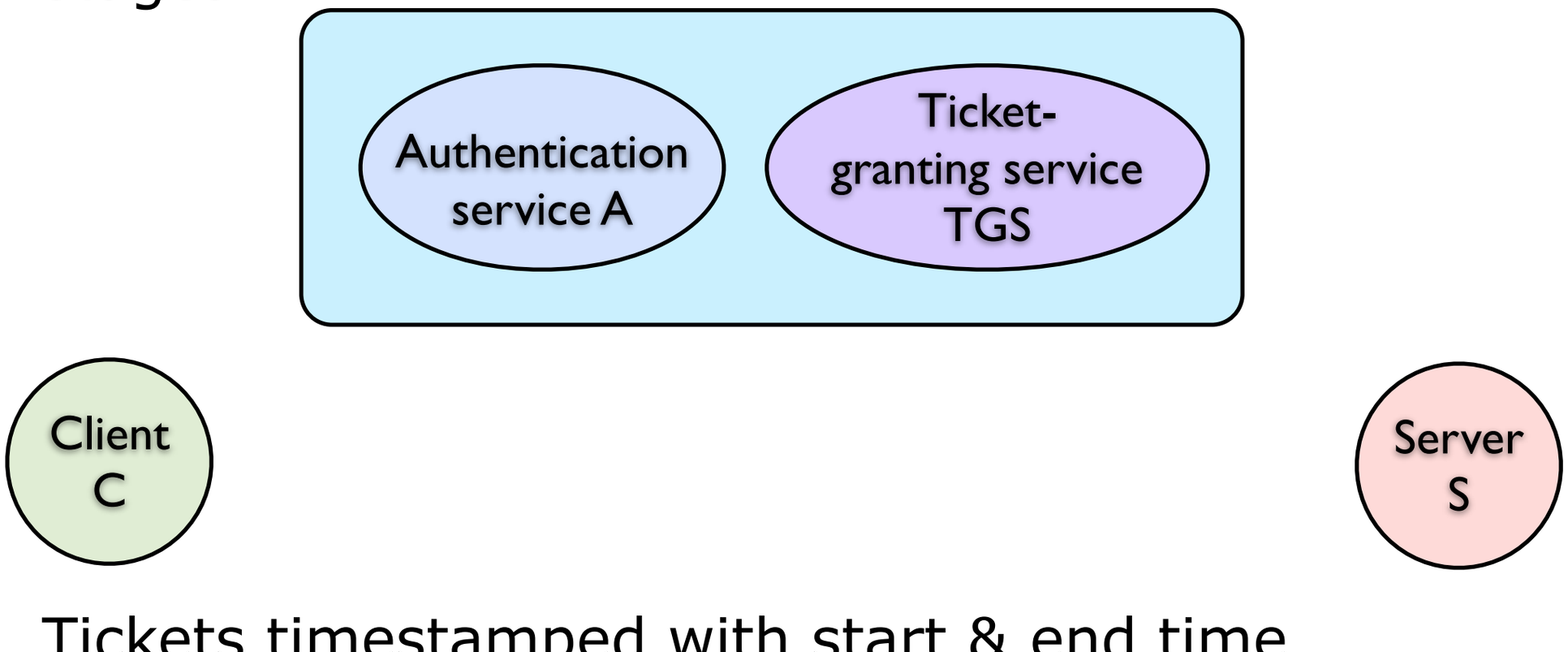
- Needham-Schoeder uses *nonces*
- Essentially, a timestamp or random unguessable value
  - used to avoid replay attacks
  - and challenge other party to decrypt/re-encrypt
- Last step uses some function f(N_B)
  - requiring Alice to prove that she was able to decrypt Bob's message using K_{AB}, do f to it, and re-encrypt

Problem

- Needham-Schroeder has a weakness
- If Mallory learns an (old but still valid) K_{AB} she can replay the third message $\{K_{AB},A\}K_B$
  - Thereby convincing Bob that she is Alice
- Can easily happen if Alice's system is compromised
- Solution: add nonce/timestamp to message 3 also: $\{K_{AB},A,time\}K_B$
  - enabling Bob to detect & reject attempts to replay stale messages

Kerberos

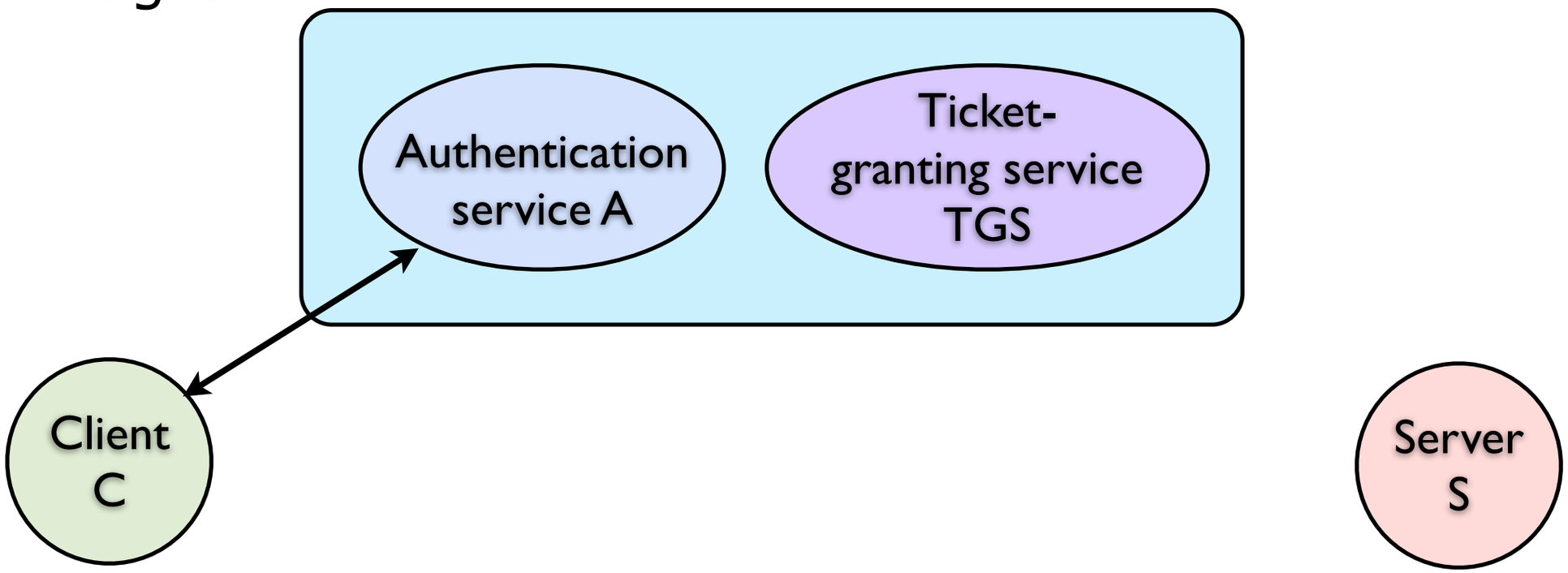
- Uses Needham-Schroeder authentication in 2 stages:



- Tickets timestamped with start & end time
  - typically valid for up to 12 hours

Kerberos

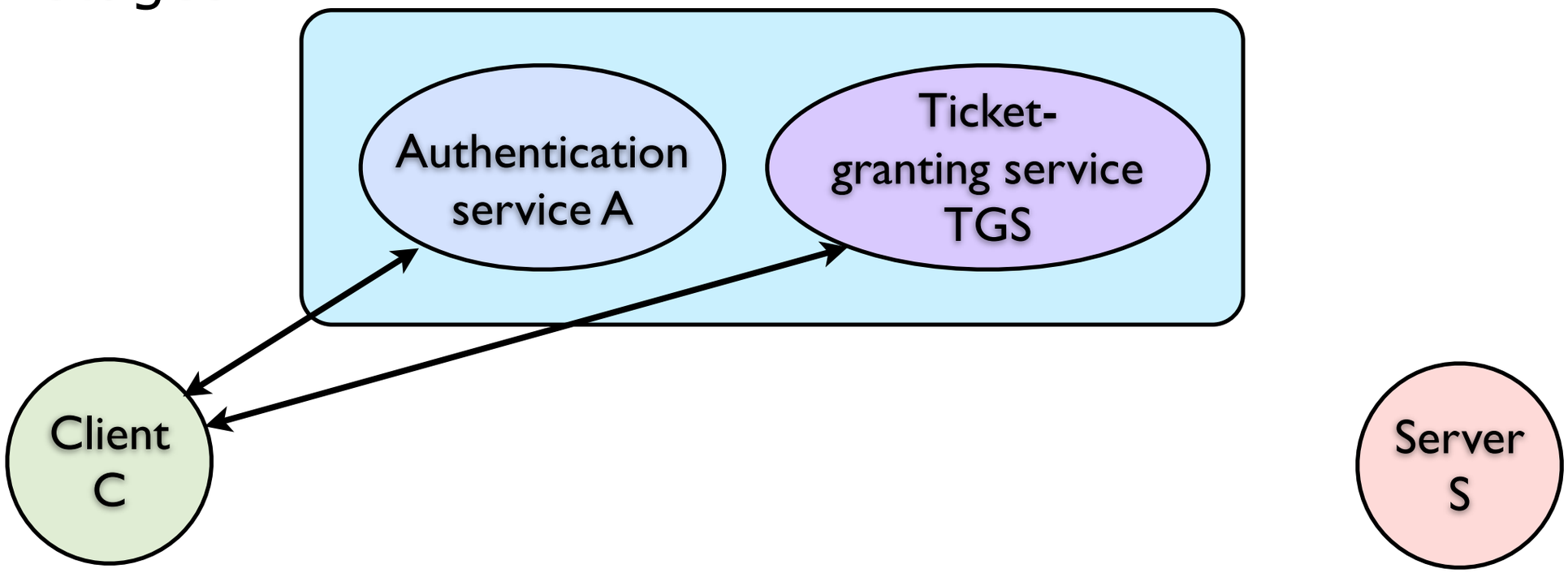
- Uses Needham-Schroeder authentication in 2 stages:



- Tickets timestamped with start & end time
  - typically valid for up to 12 hours

Kerberos

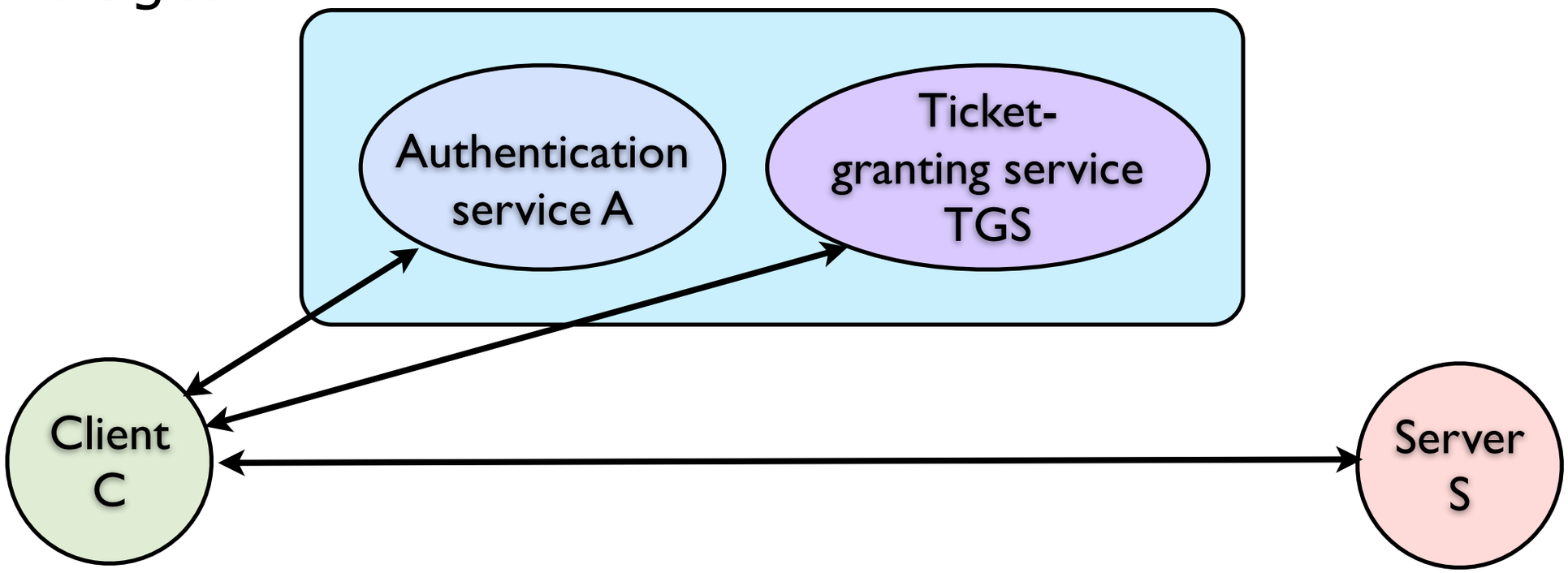
- Uses Needham-Schroeder authentication in 2 stages:



- Tickets timestamped with start & end time
  - typically valid for up to 12 hours

Kerberos

- Uses Needham-Schroeder authentication in 2 stages:



- Tickets timestamped with start & end time
  - typically valid for up to 12 hours

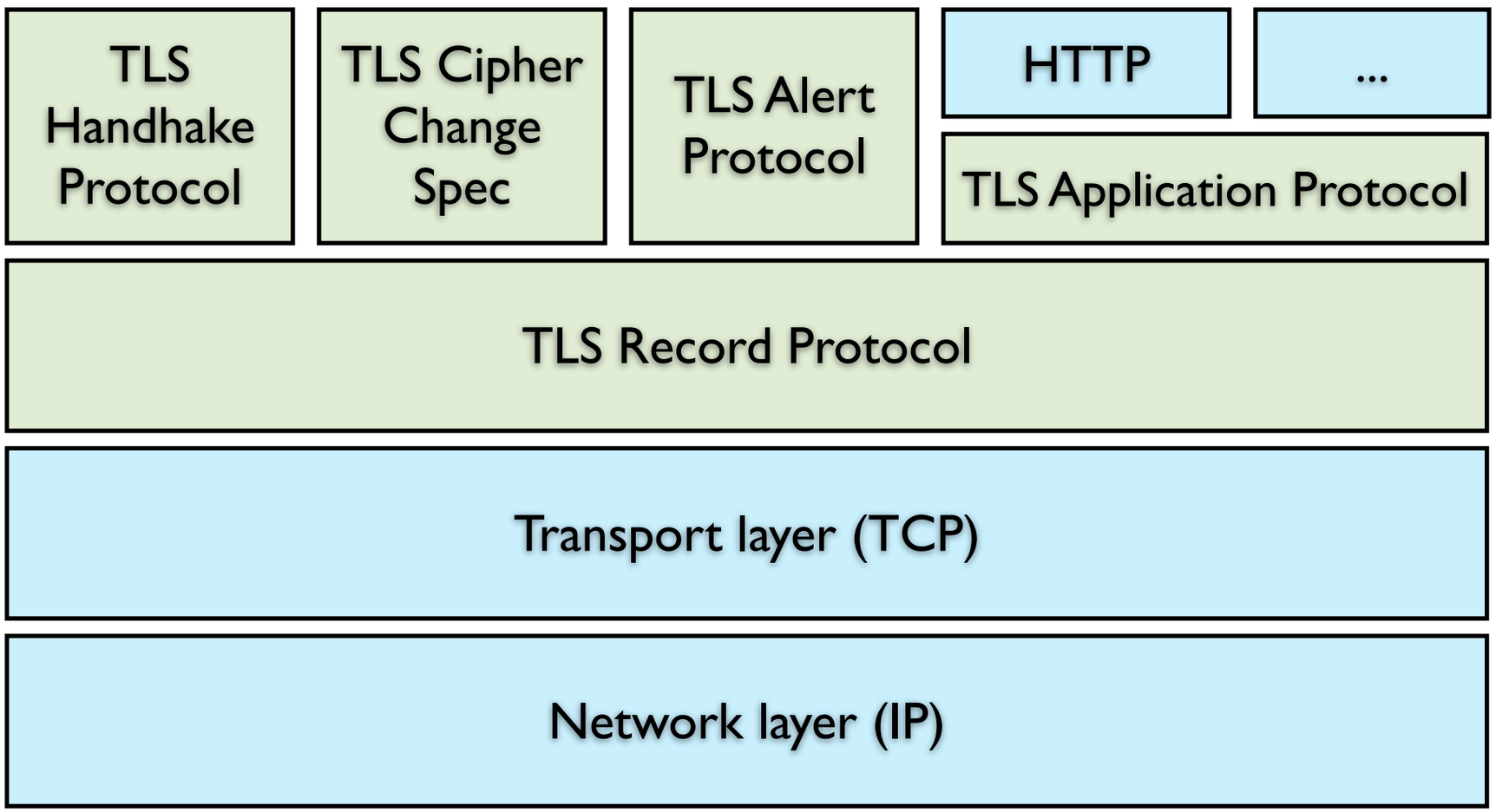
Critiques

- In earlier versions:
- Using nonces as timestamps requires clock synchronization
  - which is hard in itself
  - and may lead to indirect vulnerabilities through synchronization protocol
- Fix: allow timestamps or sequence numbers
  - Keep recent history to prevent reuse / replay
  - Difficult to guarantee in presence of server failures though

TLS

- Secure Sockets Layer (SSL)
  - introduced to support secure online transactions (e-commerce)
- Transport Layer Security (TLS)
  - extension of SSL; Internet standard RFC 2246
- Supported by most browsers
  - now used for many other applications, e.g. (secure) Web email clients

Overview


TLS Handhake Protocol
TLS Cipher Change Spec
TLS Alert Protocol
HTTP
...
TLS Application Protocol
TLS Record Protocol
Transport layer (TCP)
Network layer (IP)

Main features

- Provides a (relatively) transparent wrapper over a standard TCP/IP connection
- Negotiation to agree on protocol / key size
  - initially to avoid export restrictions, but also a good idea for futureproofing
- Use hybrid public/shared key approach to establish secure channel
- Use certificate authorities to identify server reliably
  - user is alerted if certificate doesn't match URL, or if something else goes wrong

Attacks

- TLS has a number of known vulnerabilities (cf. Wikipedia)

Survey of the TLS vulnerabilities of the most popular websites

Attacks	Security			
	Insecure	Depends	Secure	Other
Renegotiation attack	6.0% (–0.2%) support insecure renegotiation	1.4% (–0.1%) support both	85.1% (+0.5%) support secure renegotiation	7.5% (–0.3%) not support
RC4 attacks	35.7% (–0.4%) support RC4 suites used with modern browsers	56.0% (+0.2%) support some RC4 suites	8.4% (+0.4%) not support	N/A
BEAST attack	69.6% (±0.0%) vulnerable	N/A	N/A	N/A
CRIME attack	13.6% (–0.7%) vulnerable	N/A	N/A	N/A

IEEE 802.11 (WiFi)

- Wired Equivalent Privacy (WEP)
  - simple attempt to secure wireless communications
  - Access control: challenge-response (similar to Kerberos)
  - Privacy and integrity: optional encryption based on RC4 stream cipher

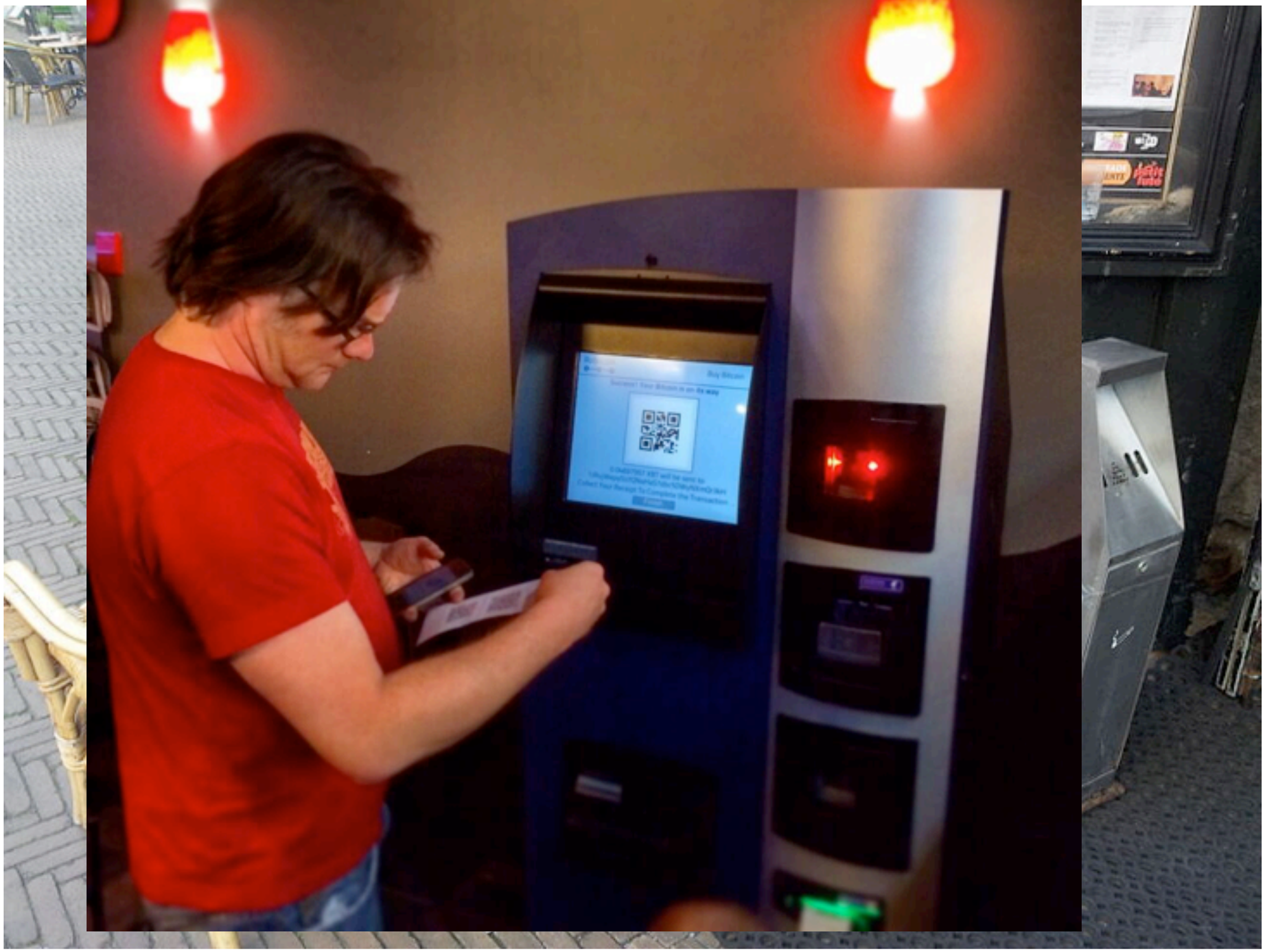
Problems with WEP

- RC4 with weak (40 or 64 bit) keys
- Stream cipher subject to brute-force attack
- Single key across network (= single point of failure)
- Base stations not authenticated (allows eavesdropping)
- Misconfiguration / lack of documentation
  - hard for end users to understand risks and how to address them
- Successors: WPA, WPA2
  - use stronger (128-bit) keys, AES encryption
  - support negotiation of cryptographic algorithms (like TLS)

Bitcoin

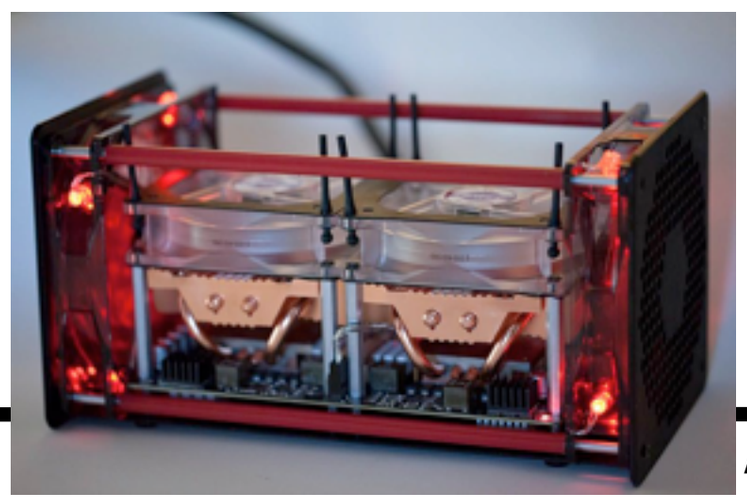
- "Cryptocurrency"
  - uses cryptography to enable creation and transfer of money
- Started in 2009
- Supported by a peer-to-peer payment network - the "Bitcoin network"
- There are now >12M bitcoins (BTCs) in circulation
  - currently each worth ~$600, over $7 billion total
  - (Arbitrary) limit of 21 million


DE WAAG
BETALEN MET
BITCOINS
MOGELIJK!
bitcoin



Bitcoin: how it works (simplified)

- Send payments by sending signed messages (public key encryption)
- Request update to public distributed database of all transactions
  - similar to BitTorrent, but stores complete history of all coins/transactions
  - transactions timestamped, forming a "block chain"
- Adding transactions to the block chain ("mining") takes work
  - This work is rewarded with more bitcoins (~ 25 BTC/block currently)
- Several different types of specialized Bitcoin mining hardware for sale

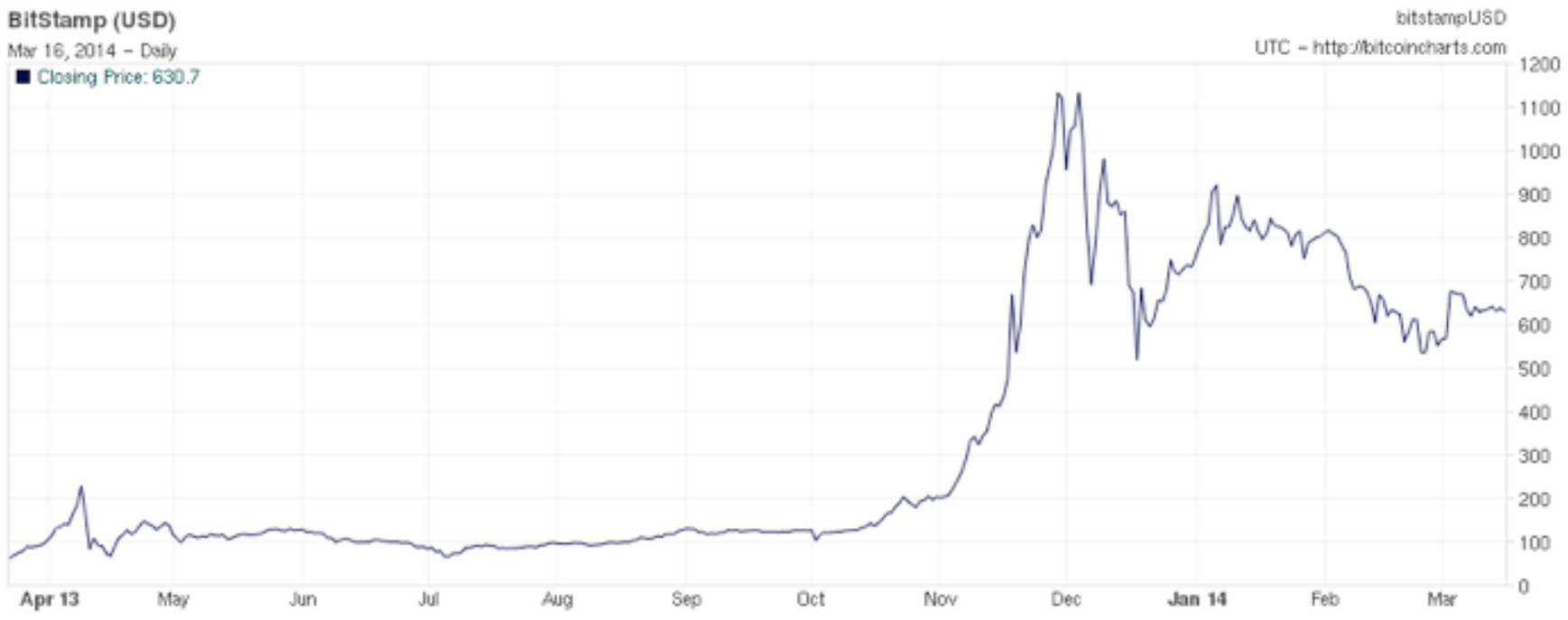


Wallets

- A collection of public/private keys
- Can be managed by:
  - Software clients (Win, Linux, Mac)
  - Websites (including currency exchange)
  - Storing electronically readable coins on paper

Is it a good idea?

- Caveat #1: volatility
  - BTC value varies a lot compared to normal currencies
  - There have been several speculative "bubbles"
  - Some people have made (or lost) a lot of (real) money investing in Bitcoin



Is it a good idea?

- Caveat #2: Private key = money
  - BTCs are no more secure than money in bank (or under mattress)
  - (Probably less, depending on how careful you are)
  - If someone learns your private key (or you lose it) there is no way to prevent theft/loss
  - "Bitcoins can be lost. In 2013 one user said he lost 7,500 bitcoins, worth $7.5m at the time, when he discarded a hard drive containing his private key." (Wikipedia)
- Caveat #3: Many governments not a big fan of Bitcoin
  - Can easily be used for criminal / black market / money laundering
  - Political/national security concerns may win out over BTC
  - What if someone (NSA) breaks PK encryption in the future?