

Myrto Arapinis School of Informatics University of Edinburgh

February 13, 2014

Credit card payment with the payment of the paymen

Credit card payment protocol

2 / 28

Behavior in the usual case



- 1. The waiter introduces the credit card
- 2. The waiter enters the amount m of the transaction
- 3. The terminal authenticates the card
- 4. The costumer enters his secret pin If the amount m is greater than 100 euros (and in only 20% of the cases)
 - 4.1 The terminal asks for authentication of the card
 - 4.2 The bank provides authentication

1/28

More details

4 actors: Bank, Customer, Card, and Terminal

Bank owns:

- ► a secret signing key sk_B
- ► a public verification key pk_B
- a secret symmetric encryption key per card K_{CB}

Card owns:

- ► Data: last name, first name, card's number, expiration date
- ► Signature's value VS = {hash(Data)}_{sk_B}
- a secret symmetric encryption shared with the bank K_{CB}

Terminal owns:

► the public verification key pk_B

5 / 28

Some flaws

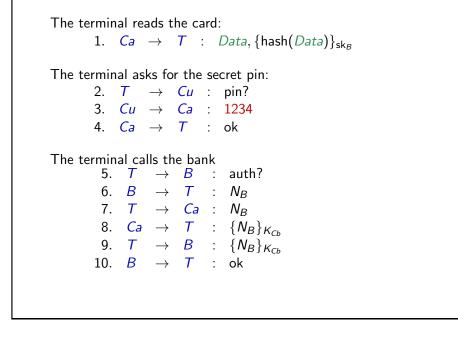
The security was initially ensured by:

- ► the cards were difficult to reproduce
- \blacktriangleright the protocol (!) and keys were secret

But:

- ▶ cryptographic flaw: 320-bit keys can be broken (1988),
- logical flaw: no link between the secret code and the authentication of the card,
- ► fake cards can be built.
- \Rightarrow "YesCard" built by Serge Humpich (France, 1998)

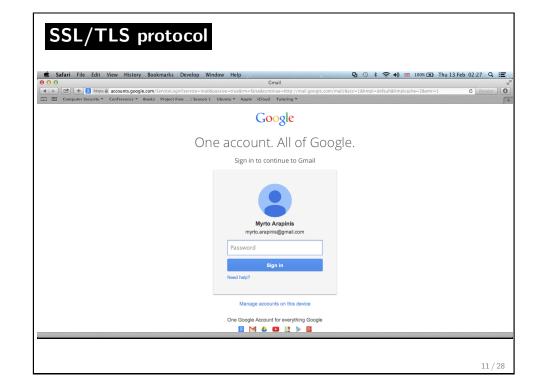
Credit card payment protocol (in short)



How does	the'	'YesC a	Card" work?	
	Ca –	→ T → Cu	: <i>Data</i> , {hash(<i>Data</i>)} _{sk_B} : pin?	
		→ Ca' : → T :	: 12345678 : ok	
There is alv \longrightarrow creatio	5			
2. 3.	Т – Си' –	→ Cu′	: 0000	

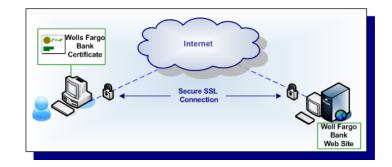
6/28

The SSL/TLS protocol	
	9 / 28



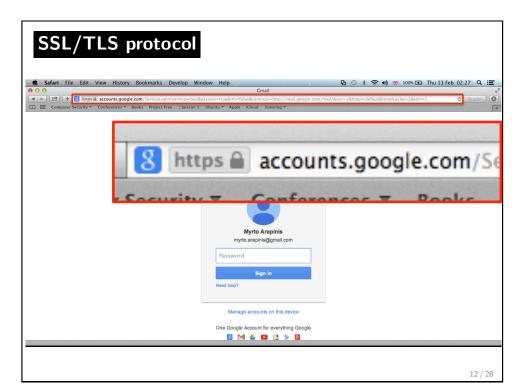
SSL/TLS protocol

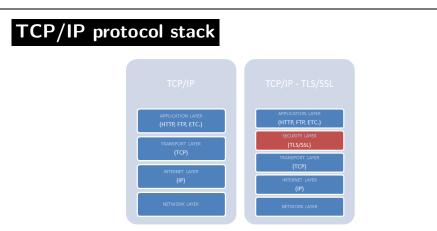
Goals: Confidentiality, Integrity, Non repudiation



SSL/TLS use X.509 certificates and hence asymmetric cryptography to exchange a symmetric key. This session key is then used to encrypt subsequent communication. This allows for **data/message confidentiality**, and message authentication codes for **message integrity** and thus, **message authentication**.

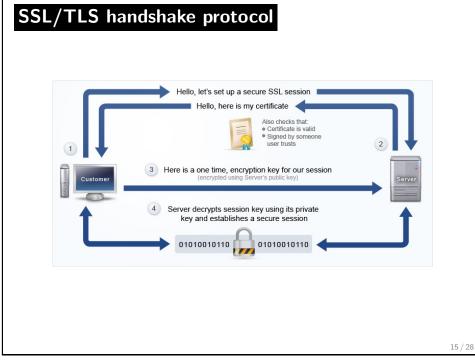
10/28



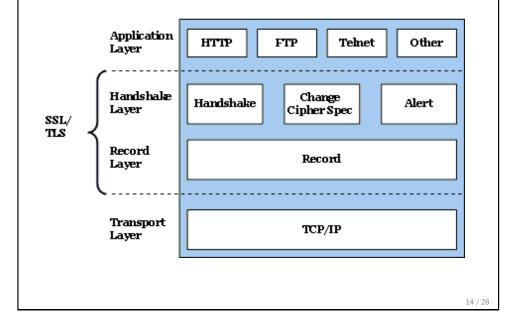


- ► TCP/IP provides end-to-end connectivity and is organized into four abstraction layers which are used to sort all related protocols according to the scope of networking involved
- ► The SSL/TLS library operates above the transport layer (uses TCP) but below application protocols

13/28



SSL/TLS protocol layers



SSL/TLS renegotiation

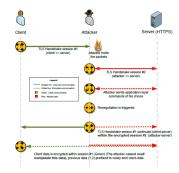
Client and server are allowed to initiate renegotiation of the session encryption in order to:

- ► Refresh keys
- ► Increase authentication
- ► Increase cipher strength
- ▶ ...

Client or server can trigger renegotiation by sending a hello meesage

SSL/TLS renegotiation weaknesses

- Renegotiation has priority over application data!
- Renegotiation can take place in the middle of an application layer transaction!



(Detailed on the board)

Incorrect implicit assumtion: the client doesn't change through renegotiation

17 / 28

Anil Kurmus' plaintext injection attack on HTTPS

Twitter status updates using its API by posting the new status to http://twitter.com/statuses/update.xml, as well as the user name and password

Attacker:

```
POST /statuses/update.xml HTTP/1.1
Authorization: Basic username:password
User-Agent: curl/7.19.5
Host: twitter.com
Accept:*/*
Content-Length: 140
Content-Type: application/x-www-form-urlencoded
status=
```

Victim: POST /statuses/update.xml HTTP/1.1 Authorization: Basic username:password...

 \Rightarrow the attacker gets the user name and password of the victim!

Marsh Ray's plaintext injection attack on HTTPS

Attacker:

GET /pizza?toppings=pepperoni;address=attacker_str HTTP/1.1
X-Ignore-This:(no carriage return)

Victim:

GET /pizza?toppings=sausage;address=victim_str HTTP/1.1 Cookie:victim_cookie

Result:

GET /pizza?toppings=pepperoni;address=attacker_str HTTP/1.1
X-Ignore-This:GET /pizza?toppings=sausage;address=victim_str HTTP/1.1
Cookie:victim_cookie

 \Rightarrow Server uses victim's account to send a pizza to attacker!

18 / 28

The SAML Signle Sign On (SSO) protocol

SAML SSO protocol

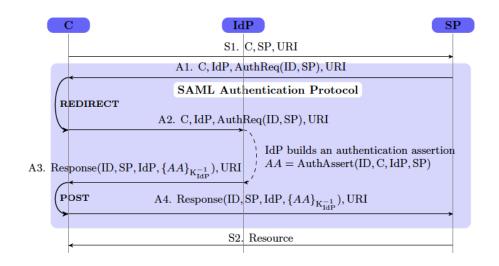
Statewood Statewood Statewood		🖸 Sign in	News	Canad	Weather	iPlayer	TV	Radio	More -	Search	Q	 \$
	BBC	🕞 əign in	News	Sport	weather	inayer	14	readio	more •	Search	ų	
Al Liverbool Al Liverbool A												
As Duepool As a				A			BALL					
 UK parties will social social	Storm upor latest for	dates: Get the where you are	in	Con the		Triple I guilty	killer's	accomp	olices F	ulham 2-3	Liverpool	
Air to easier to them if your of London 2012* Image: Construction of the provided in the provid	6			60		= UK partie = Comedia = Carney a	n Sid Cae djusts int	erest rates	91 = policy =	Decisive day fo Collingwood ha Arsenal 0-0 Ma	nded England role nchester United	
BBC FAEL COMEDY BBC FAED 2 FACTUR. David Attenborough and th ** Audiend boy's parents collect MBE ** Audiend boy's parents collect MBE ** + Audiend boy's parents collect MBE ** + Hodgson rules out Teny return ** + Hodgson rules out Teny return	D		■))	Real Property lines	1	= Man plea	vs LOND ids guilty	DN to Rigby vi	deos =	OOTBALL Weather hits P		
BBC now Entertainment News Lifestyle Knowledge Sport	Outnumbere				and th				t MBE	Newcastle Unit Hots		
		BBC now E	ntertainmen	t	News	Life	style	Kn	owledge	Sp	ort	
s://ssl.bbc.co.uk/iid/sionin												

O DBB BBC - Sign in ×	2						
C https://ssl.bbc.co.uk/id/	BBC Q	lign in Nev	rs Sport Weather	r iPlayer TV	More -	Search Q	
		I IN BBC i					
	Email or usern Password Remember m	ID? Please register. ame Forget your password e Untok if you're using a c Cancel			Google About BBC Simple Register quickly, favourites, and r Safe We store your in never shore it will Spam-free	BBC for 30 days. coge you are 16 or over. your activity to Facebook or D and easily to comment. add	
	BBC iWọnder		10 dl	How did Pac	:k Up Your	Gareth Malone explains why	
	a set	Smil	1	viral hit of W		the song was a success	
	News	Sport	Weather	iPlayor	TV	Radio	



SAML SSO protocol (OASIS 2005)

SAMI SSO protocol

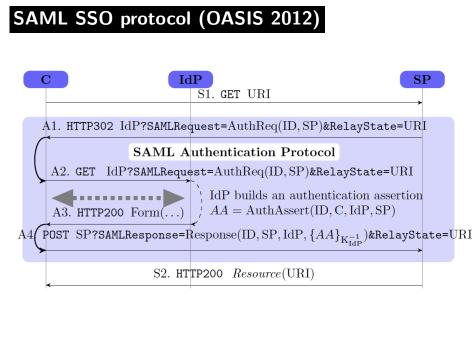


Google's implementation of SSO

Google's SAML-based Single Sign-On for Google Applications deviates from the above protocol for a few, seemingly minor simplifications in the messages exchanged:

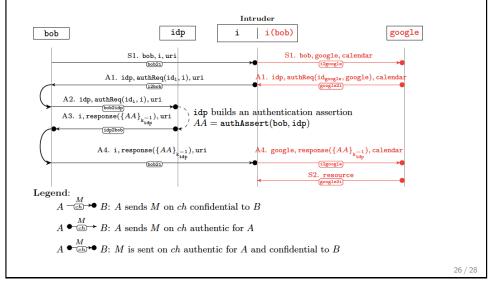
- G1. ID and SP are not included in the authentication assertion, i.e. AA = AuthAssert(C; IdP) instead of AuthAssert(ID; C; IdP; SP);
- G2. *ID*, *SP* and *IdP* are not included in the response, *i.e.* $Resp = \text{Response}(\{AA\}_{K_{IdP}^{-1}})$ instead of $\text{Response}(ID; SP; IdP; \{AA\}_{K_{IdP}^{-1}}).$

25 / 28



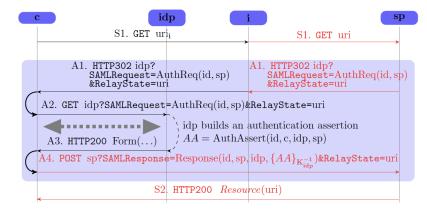
Attack Google's SSO implementation

[A. Armando, R. Carbone, L. Compagna, J. Cullar, L. Tobarra, "Formal analysis of SAML 2.0 web browser single sign-on: breaking the SAML-based single sign-on for google apps", (FMSE'08)]



Attack SAML SSO protocol (OASIS 2012)

[A. Armando, R. Carbone, L. Compagna, J. Cullar, G. Pellegrino, A. Sorniotti, "From Multiple Credentials to Browser-Based Single Sign-On: Are We More Secure?", Chapter in Future Challenges in Security and Privacy for Academia and Industry]



\Rightarrow XSS attack on SAML-base SSO for Google Apps