Advanced Web Security - Assignment 1A

Oliver Nilssen - ol0716ni-s

November 2020

Questions

1. Briefly explain how double-spending is prevented in Bitcoin. How can an attacker with huge computational resources perform double-spending?
2. When Alice buys something from Bob using the untraceable E-cash scheme, why is it impossible for Bob to learn the identity of Alice?
3. In the untraceable E-cash protocol in the lecture notes, the serial number of a coin is a signature from the bank, i.e., produced using the bank's private key. Why (in a technical sense) can the bank not map this serial number to Alice?
4. When saving a payment method to a merchant website / app, what are the security implications of storing card's data in encrypted form?
5. In EMV SDA, why is it important that the Certification Authority signs the card's public key? What would differ if CA signs the card's secret key instead?
6. Describe an attack that the CVV1 code on a credit card prevents. Why is it not effective against skimming?
7. When requesting a blind signature, why must Alice keep r secret?
8. Give examples of a feature and a 'bug' that derive from the multiplicative property of RSA digital signatures.

Answers

1. Bitcoin relies on a universal ledger called a blockchain. The blockchain provides a way for all nodes to be aware of every transaction made and with bitcoin all transactions are publicly announced to all nodes. The way to solve double-spending with bitcoin is that there will be multiple of nodes working on different transactions, and if you double-spend by sending two transactions at the same time, the transaction that has had the most work on it will usually be the one that is agreed on by the other nodes to be the "most real" one, and all other work will be discarded. Which means only one of the two transaction you sent through will be legit and the other one will be discarded.
   The way for huge computer resources can perform double-spending is by having 51% of the mining pool all by themselves. The reason this is possible is because that is the amount of estimated "mining" power that is needed to mine a new node onto the blockchain, which means they can do a reverse transaction and perform double spending

2. Bob can't learn the identity of Alice because her coins are signed using blind signatures. Neither the bank or the receiver of the coin can tell where it is from. The coins are signed from the bank, but they don't know who they signed the cash for. Alice only sends Bob the coin and the Signature which the bank produced, but as mentioned above the bank never got the information about who they signed the coin for.

3. Alice picks a random message 'r' which is used in order to blind the signature, so the bank doesn't know what it is signing. She takes the value 'r' and multiplies it by the hash of the coin. This value is B, the blinded message. The bank then receives the message B from the user, which the Bank then signs using it's secret key - d and then returns the $\bar{S}$. When Alice gets $\bar{S}$ she then can extract the signature of x (or hash of x) by multiplying $\bar{S}$ by 'r' inverse (mod n). This will give Alice a valid signature of the coin, but the bank has never seen x or hash of x, so it can never trace where it originated or where it is spent.

4. The security implications of storing card's data in encrypted form, When saving a payment method to a merchant website/app is with the use of encryption itself, there is one single secret key that can decrypt any of the credit-cards stored on file or any cipher text during payment. It means that there is a single point of failure that is very vulnerable to attacks. If the secret key used to encrypt all credit-cards is stole, then as mentioned all credit-cards that is saved will be reveled.

5. In EMV SDA (static data authentication), it is important that the Certification Authority signs the card's public key because we need to testify that the card belongs to a specific issuer that issued the public key. Which means the bank has registered the public key of your card to Visa or MasterCard.

If the CA signs the card's secret key the private key would no longer be secret. It is not supposed to be signed, as that is what the public key is for, to be public. Exposing the private key would mean that potential malicious actors could try and get the key while it is being signed, which means they will be able to encrypt and decrypt all future messages. The private key does not hold the information to confirm the identity of the issuer either, so it wouldn't be a valid signature.

6. CVV1 can help against card-not-present attacks (eg. online shopping), where the attacker might have gotten the credit card details from a skimmer or something similar, but is missing the CVV1 code on the back of the card. The CVV1 will not be included when you use your physical card in a store or on an ATM. The CVV1 is neither punched into the any physical machine and it is not available in the magnetic strip of the card. This is also why the CVV1 is not effective against skimming, as the card can still be skimmed and the data can be printed onto an empty card and reused. But it will be harder to use this card for online purchases as it will require the CVV. The code is a MAC based on the account number, version number and expiry date. The CVV is as an extra form of security against actors using your card for card-not-present transactions.

7. Alice must keep r secret because if the other party know 'r' then they are able to un-blind the message and get the identity of Alice by using the inverse or 'r'. 'r' is the secret that helps Alice blind the message in the first place. Having it reveled would be just as telling everyone who the coin belongs to in the first place.

   A bug that derive from the multiplicative property of RSA digital signatures has to do with the fact that we can multiply two signatures which will create a new signature for a new message which is a multiplications of the two previous messages.

$$\begin{aligned} S_1 \cdot S_2 &= x_1^d \cdot x_2^d \; mod \; n \\ &= (x_1 \cdot x_2)^d \; mod \; n \end{aligned}$$

   Here we can see that $x_1 \cdot x_2$ is a new message. Which the signer never signed for, but it is a valid signatures that is derived from this multiplication. An attacker could mount this attack by using arbitrary data in a blind signature to get the needed signature. We can prevent this by hashing the message

   A feature of this property in RSA signatures is the ability to sign signatures blindly. This allows the signatures to be valid even though the signer never saw the actual message of what it was signing. This is done using a random value 'r' to the power of the public verification key, which is then timed by the hashed version of the message (or key in this instance). This can then be used to obtain a hash of the hashed coin.