Case Study: Virtual Election

1.Is it virtually possible to have elections on the internet? How? What sort of infrastructure would be needed for this?

With the changing world of computer networks, many people can now access them over the Internet, making electronic voting a feasible option for conducting elections. The main problem would be ensuring the same level of security as regular elections. To resolve the security difficulties in the election process, a variety of cryptographic algorithms can be applied.

Surprisingly, conducting such elections does not require much. A number of websites, such as Election Runner, have been developed in recent years that allow anyone to conduct secure virtual elections. Virtual elections can also be conducted using a variety of real-time meeting platforms, such as Google Meet and Zoom, as well as common survey platforms, such as Google Forms. So all you will need is a voting device and an account with the platform. It's really that simple.

2.What would be the main concerns in such a virtual election?

Security is always the primary concern in virtual elections. As a result, it's best to hold the election on a platform that has been designed specifically for the work, such as Electron Runner. It uses secure voting, with each voter having their own "Voter ID" and "Voter Key" and only being able to vote once. Additionally, all elections are secured with SSL grade security and 256 bit encryption, ensuring the security of your election and ballots. There's no way the voting is unsafe unless your account has been hacked.

Another issue to consider is voter’s education. Not everyone, — especially the elderly, is familiar with or comfortable with the concept. Electronic voting, on the other hand, is a distinct possibility in the coming years.

3.What would be the use of digital signatures and encryption in virtual elections?

A digital signature is a mathematical technique for verifying the integrity and authenticity of a message, software, or digital document. It's the digital equivalent of a handwritten signature or a stamped seal, but it has a lot more security built in. The purpose of a digital signature is to prevent tampering and impersonation in digital communications. Electronic documents, transactions, and digital messages can all profit greatly from digital signatures as proof of origin, identity, and status. They can also be used to acknowledge informed consent by signers.

An encryption technique, such as RSA encryption, is used in electronic voting. It's an asymmetric cipher that encrypts and decrypts data using two different keys, the public and private. RSA encryption is used to create the digital signature. The hashing of the message generates a message digest, which is the first step in the process. The signature is then created by

encrypting the digest with the sender's private key. The receiver will hash the message using the same digest function to verify it. The signature is decrypted using the receiver's public key at the same time. After that, the outputs of the two processes are compared. If they are equal, the message is authenticated, and the message's integrity is preserved.

Case Study: Contract Signing

1.When can a contract in real-life be considered to be complete?

In an ideal world, a contract is considered complete when all of the contract's obligations have been fulfilled and the remaining costs and risks are minimal. This signifies that all contract parties have completed their obligations as specified in the contract and as agreed upon at the time the contract was drafted. This is clearly the best option, as it indicates that the contract has been completed in its entirety and is only being terminated because the agreed-upon activities have been completed as required.

2.Is it sufficient if only one of the parties signs a contract?

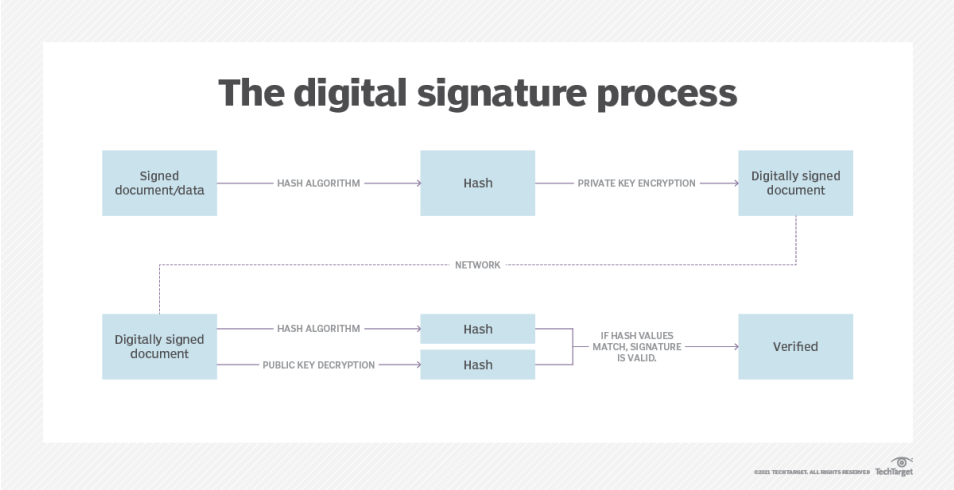
No, it isn’t sufficient for only one party to sign a contract. It must have all the parties involved in the process to sign the contract document. Every party must agree to the terms and conditions on the contract and are obligated to follow the contractual duties.

3.What mechanism can be used in cryptography to sign electronic contracts?

Digital signatures, digital certificates, and advanced electronic signatures are all types of electronic signatures that use cryptography to authenticate the identity of the person signing and to secure their electronic signature.

Public-key infrastructure (PKI) is a popular form of cryptography that digital signatures and digital certificates often use. PKI requires that the contract provider and the signer agree on a shared secret key that no one else knows.

When a signer electronically signs a document, the signature is created using the signer’s private key, which is always securely kept by the signer. The mathematical algorithm acts like a cipher, creating data matching the signed document, called a hash, and encrypting that data. The resulting encrypted data is the digital signature. The signature is also marked with the time that the document was signed. If the document changes after signing, the digital signature is invalidated. To protect the integrity of the signature, PKI requires that the keys be created, conducted, and saved in a secure manner, and often requires the services of a reliable Certificate Authority (CA).



4.How can disputes be settled if a party later refuses having signed a contract?

An agreement by the parties to submit to arbitration all or certain disputes which have arisen or which may arise between them in respect of a defined legal relationship while signing the contract agreement is called atribation. In this process, a third party has the contractual document signed between the two parties which can be shown while disputes like this arise later after having signed a contract.

Case Study: Cross Site Scripting (XSS) Vulnerability

1. **What is the purpose of scripting technologies on the internet?**

   A scripting language is a programming language for a runtime system that automates the execution of tasks that would otherwise be done by a human operator individually. Scripting languages are typically interpreted rather than compiled at runtime. Some of the uses of scripting language are discussed below:

   - Scripting languages are used in web applications. It is used in server side as well as client side. Server side scripting languages are: JavaScript, PHP, Perl etc. and client side scripting languages are: JavaScript, AJAX, jQuery etc.
   - Scripting languages are used in system administration. For example: Shell, Perl, Python scripts etc.
   - It is used in games applications and multimedia.
   - It is used to create plugins and extensions for existing applications.

2. **What can prevent CSSV attacks?**

Cross-site scripting (also known as XSS) is a web security vulnerability that allows an attacker to modify how users interact with a vulnerable application. It allows an attacker to get around the same origin policy, which is meant to keep websites separate from one another. Cross-site scripting vulnerabilities allow an attacker to impersonate a victim user and perform any actions that the user is capable of, as well as access any of the user's data. If the victim user has privileged access to the application, the attacker may be able to take complete control of the app's functionality and data.

Preventing Cross-site Scripting (XSS) is not easy. Specific prevention techniques depend on the subtype of XSS vulnerability, on user input usage context, and on the programming framework. However, there are certain general strategic principles that you should follow to keep your web application safe. There are the general steps you could carry out to prevent XSS attacks:

Step1: Train and maintain awareness

To keep our web application secure, everyone involved in its development must be aware of the dangers of XSS vulnerabilities. All developers, QA staff, DevOps, and SysAdmins should receive appropriate security training.

Step2: Don’t trust any user input

Consider all user input as untrusted. Any user input that is used in HTML output exposes the user to the risk of an XSS. Treat input from authenticated and/or internal users the same way you treat input from the general public.

Step 3: Use escaping/encoding

Depending on where user input is to be used, use the appropriate escaping/encoding technique: HTML escape, JavaScript escape, CSS escape, URL escape, and so on. If you need to escape, use existing libraries rather than writing your own.

Step 4: Sanitize the HTML

You can't escape/encode user input that contains HTML because it would break valid tags. Use a trusted and verified library to parse and clean HTML in such cases. Choose a

library based on your programming language, such as HtmlSanitizer for.NET or SanitizeHelper for Ruby on Rails

Step 5: Set the HTTP Flag only

Set the HttpOnly flag for cookies to mitigate the effects of a possible XSS vulnerability. If you do, client-side Javascript will not be able to access such cookies.

Step 6: Use a Content Security Policy

Use a Content Security Policy to help mitigate the effects of a possible XSS vulnerability (CSP). CSP is an HTTP response header that enables users to declare which dynamic resources are permitted to load based on the request source.

Step 7: Scan regularly

XSS vulnerabilities can be introduced by developers or by third-party libraries, modules, and software. We should use a web vulnerability scanner to scan our web applications on a regular basis.

3. **What sort of testing can the creator of a website perform in order to guard against possible CSSV attacks?**

Cross-site Scripting attacks (XSS) can be used in a number of ways by attackers to compromise application security. It's most commonly used to steal session cookies, allowing the attacker to pretend to be the victim. Furthermore, XSS vulnerabilities have been exploited to create social media worms, spread malware, deface websites, and phish for credentials. They've also been used in combination with social engineering techniques to escalate to more damaging attacks like stealing personal information.

Stored XSS, Reflected XSS, and DOM-based XSS are the three major categories of cross-site scripting.

Stored XSS are all about getting a malicious parameter reflected to the user. So the straightforward approach to avoid XSS vulnerabilities is to sanitize user data and handle inputs safely.

From the user's point-of-view, vigilance is the best way to avoid XSS scripting. This means not clicking on suspicious links which may contain malicious code. Additionally, web application firewalls (WAFs) also play an important role in mitigating reflected XSS attacks. With signature based security rules, supported by other heuristics, a WAF can compensate for the lack of input sanitization, and simply block abnormal requests.

The primary rule that you must follow to prevent DOM XSS is: sanitize all untrusted data, even if it is only used in client-side scripts. If you have to use user input on your page, always use it in the text context, never as HTML tags or any other potential code.

References

1. What is a Digital Signature?
2. How Digital Signatures Work.
3. Secure E-Voting With A Blind Signature
4. Election Runner: Build a Secure Online Election for Free
5. The Important Uses of Cryptography in Electronic Voting and Counting
6. Introduction to Scripting Languages
7. What is Cross-site Scripting and How Can You Fix it?
8. How to Prevent Cross-Site Scripting (XSS) Attacks