Advisory on the

SECURE DEVELOPMENT AND PROVISION OF DISTRIBUTED LEDGER TECHNOLOGY-ENABLED SERVICES

CSA SINGAPORE

VERSION 1.2

Cyber Security Agency of Singapore (CSA)

In consultation with:
Monetary Authority of Singapore (MAS)
Bank of America N.A.
Citibank N.A.
Deutsche Bank
J.P Morgan
United Overseas Bank Limited
UBS AG
BHOP Consulting Pte. Ltd.
DBS Vickers Securities (Singapore) Pte. Ltd.
Digital Treasures Center Pte. Ltd.
Fomo Pay Pte. Ltd.
Independent Reserve SG Pte. Ltd.
Paxos Global Pte. Ltd.
Triple A Technologies Pte.Ltd

Contents

NOTICE

This document is intended to guide DLT users on safe adoption, raise cyber hygiene and resilience of DLT-enabled services.

This is a living document that will be subjected to review and revision periodically. This document aims to recommend best practices for the management of risks arising from the usage of DLT-enabled services. Organisations using DLT-enabled services, service providers and developers should assess the suitability of this document for their intended use and are encouraged to consider how they can apply the recommended best practices to their specific circumstances and seek professional advice, where required. Users should also consider whether additional measures are required to secure their usage of DLT-enabled services and exercise their professional judgement if and when implementing these best practices.

Adherence and the implementation of recommendations in this document does not exempt organisations using DLT-enabled services, service providers and developers from any legal obligations and other regulatory requirements.

Subject to the maximum extent permitted under law, CSA shall not be liable for any errors and/or omissions contained herein or for any losses or damages (including any loss of profits, business, goodwill, or reputation, and/or any special, incidental, or consequential damages) in connection with the use of this material. Where in doubt, organisations using DLT-enabled services, service providers and developers are advised to obtain their own legal and/or technical advice in relation to the contents and/or implementation of the recommendations in the document.

1. Introduction

Distributed Ledger Technology (DLT) refers to a technological approach that keeps records of transactions (ledger) shared across a set of distributed networks of computer server (nodes) and synchronised among DLT nodes using a consensus mechanism[1]. Blockchain is a specific type of DLT comprising digitally recorded data that are arranged as successively growing chain of blocks, with each block cryptographically linked and hardened against tampering and revision[2]. DLT-enabled services have seen adoption in financial and non-financial domains ranging from digital currencies, digital identities to supply chains. The core value proposition of DLT is that it assures the security of data records, based on attributes of immutability, tokenisation, decentralisation, distribution and encryption, without the need for a central authority.

[1] Source: ISO 22739:2020 Blockchain and distributed ledger technologies — Vocabulary
[2] Source: Distribute Ledger Technology Terms and Definition, ITU-T

2. Purpose, Scope, and Intended Audience of this Document

This document provides best practices on applying security-by-design and mitigating measures to securely develop and use DLT-enabled services (Public and Private DLT systems), referencing international standards from International Organisation for Standardisation (ISO), National Institute of Standards and Technology (NIST) and European Telecommunications Standards Institute (ETSI).

The intended audience of this document include, but are not limited to, the following:

- Organisations using DLT-enabled services; and
- DLT service providers/developers of DLT-enabled services.

3. DLT Reference Architecture

ISO 23257 "Blockchain and Distributed Ledger Technologies (DLT) - Reference Architecture" describes a reference architecture of DLT systems (see Figure 1). The reference architecture addresses concepts, cross-cutting aspects, architectural considerations, and views, including functional components, roles, activities, and their relationships for DLT. It provides the design framework and guidance for DLT service providers to securely deploy DLT platforms.

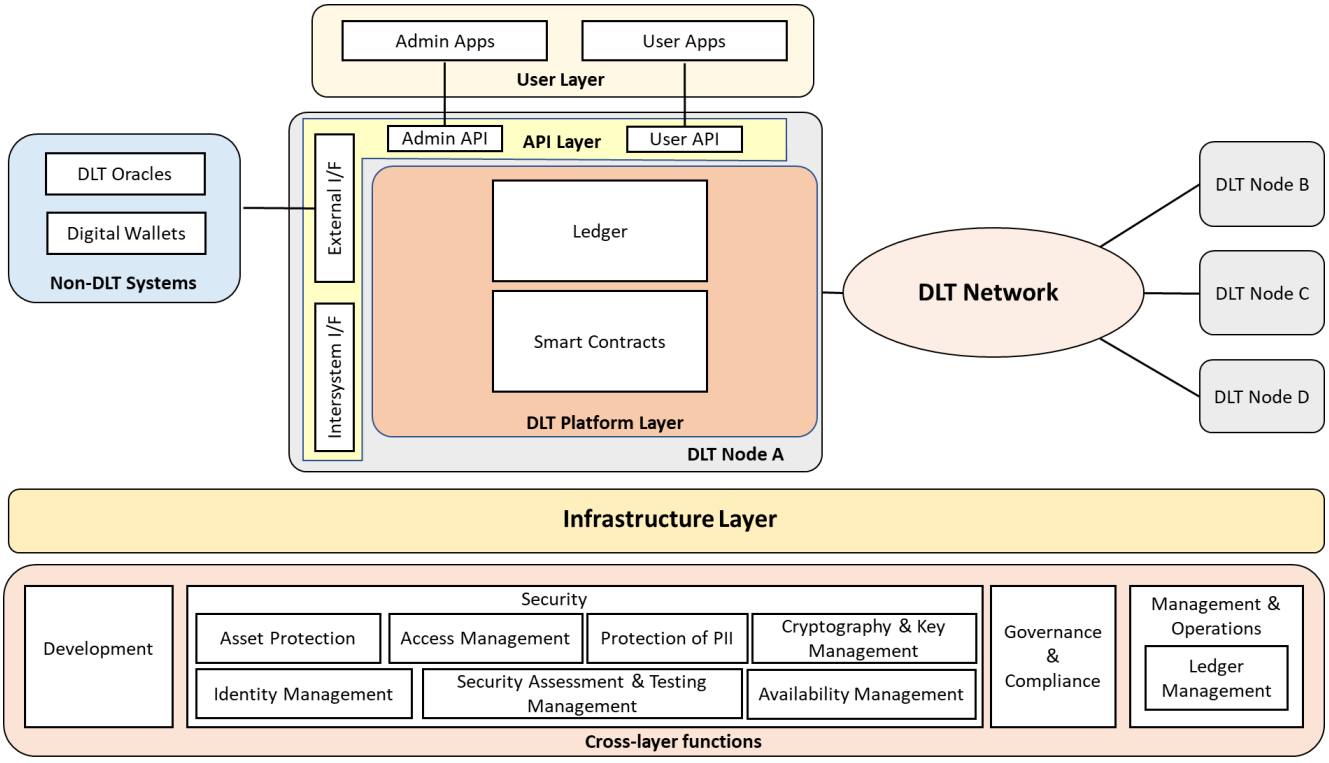

Figure 1. DLT Reference Architecture [3]

The DLT reference architecture consists of **User, API, DLT Platform** and **Infrastructure layers, DLT networks, non-DLT systems** as well as a set of **Cross-layer functions**:

a. The **User layer** contains functions to enable DLT customers to interact with other layers and cross-layer functions;

[3] Adapted from ISO 23257:2022 – Blockchain and Digital Ledger Technologies – Reference Architecture

b. The **API layer** executes functions that deliver dependable and effective access to the DLT system by calling functional components in the DLT Platform layer, enabling access to the underlying services of the platform layer and the cross-layer functions;
c. The **DLT Platform layer** comprises core functions of DLT systems that can run in a DLT node. It also connects hardware or network infrastructure provided by the infrastructure layer to relevant services in the API layer;
d. The **Infrastructure layer** provides the operating environment including networking, computation and storage components required for the normal operation of a DLT system;
e. The **DLT network** is a network of DLT nodes that make up a distributed ledger system and communicate via peer-to-peer networks;
f. **Non-DLT systems** contain functional components outside the DLT system that the DLT system communicates with to achieve its business goals; and
g. The **Cross-layer functions** support the components across all the functional layers.

4. Global DLT Threat Landscape

Given the strong economic potential of this emerging technology globally, cyber criminals have been increasingly exploiting DLT security vulnerabilities. An analysis of DLT incidents revealed that cyber criminals had stolen over US$7 billion through 290 DLT incidents between Jan 2011 and Jul 2022[4]. The most common modus operandi involves exploiting smart contract vulnerabilities and stealing private keys of wallets, with attackers targeting decentralised applications (DApps) that run on public DLT networks such as Ethereum, Polygon and EOS.

In Feb 2022, an attacker discovered a smart contract bug in a DeFi platform, Wormhole, and forged a valid signature for a transaction, stealing US$326 million. In Dec 2021, the decentralised finance platform Grim Finance lost US$30 million after an attacker exploited a common vulnerability that involved functions of the smart contract being called repeatedly (known as re-entrancy attacks) before the first invocation of the function was completed, causing loss of funds through repeated execution of functions in the smart contract[5]. In 2018, attackers breached the Coincheck cryptocurrency platform, stealing users' private keys and making off with US$500 million worth of tokens. The scale of losses is poised to significantly increase in tandem with the mainstream adoption of DLT and proliferation of digital payment tokens across many sectors.

From the analysis of DLT threat landscape reports and major DLT incidents from 2011 to 2022 (Table 1 and Table 2 in Section 10), the most common DLT attacks pertained to **security breaches** and **DeFi breaches** that took advantage of vulnerabilities in **cryptography and key management**, **smart contracts** and **digital wallets**.

[4] **Source:** Crypto & DeFi Hacks & Scams Report 2021, **Crystal Blockchain**

[5] Re-entrancy attacks are one of the most common disruptive attacks on DLT systems and applications. **Source:** List of well-known attacks, **Consensys**

DLT Threat Landscape Report	DLT Attack	Corresponding DLT Component
McAfee: Blockchain Threat Report	Security Breach[6]	Digital Wallets
		Cryptography & Key Management
Cloud Security Alliance: Top 10 Blockchain Attacks, Vulnerabilities & Weaknesses 2021	Security Breach	Digital Wallets
		Cryptography & Key Management
	DeFi Breach[7]	Smart Contracts
	51% Attack	DLT Platform Layer
Crystal Blockchain: Crypto & DeFi Hacks & Scan Report 2021	DeFi Breach	Smart Contracts
	Security Breach	Digital Wallets
	Wallet Providers Breach	Smart Contracts
		Digital Wallets
		Cryptography & Key Management
Chainalysis: The 2022 Crypto Crime Report[8]	DeFi Breach	Smart Contracts
	Security Breach	Digital Wallets
		Cryptography & Key Management

Table 1. Analysis of Common Attacks from DLT Threat Landscape Report

Based on the DLT threat landscape and incidents, cryptography and key management, smart contracts and digital wallets typically faced the most number of attacks compared to the rest of the components. The common motivation of attackers is to steal digital payment tokens (DPT) either through vulnerabilities in smart contracts or digital wallets. For a start, security vulnerabilities relating to cryptography and key management, smart contracts as well as digital wallets should be prioritised for remediation (See sections 5, 6, 7 and Annexes A, B for more details on these components). Subsequent versions of the advisory would explore the inclusion of other components in due course.

[6] Security breaches may involve phishing, malware, social engineering attacks and exchange hacks
[7] DeFi breaches may involve code exploitation, flash loans and exit scams
[8] Based on total digital payment tokens stolen by victim type

5. DLT Security at the Systems Level

DLT systems are reliant on consensus mechanisms, cryptography management, distributed nodes and communication for the execution of transactions and exchange of information. Security attributes such as **confidentiality**, **integrity**, **availability**, **non-repudiation, authentication**, **authorisation**, and **access control** [9] are necessary for the assessment of DLT systems. Through a **security-by-design** approach, these attributes could be considered upfront during the design or procurement of DLT systems.

Governance frameworks and mechanisms can bolster **security through risk-prioritised controls** to defend against known and emerging threats, **ensure vigilance through threat intelligence, raise situational awareness** to detect harmful behaviour and **enhance resilience to recover from and minimise the impact** of cyber incidents. The functions of DLT systems that address these attributes include:

a. Ledger management. DLT systems should ensure proper handling, logging and traceability of internal and external events in order to ensure non-repudiation, auditability and transparency. Recorded information about events should be tamper-proof, regardless of whether an event is recorded on-ledger or off-ledger. Where modifications are needed, this can be done by adding new records while preserving previous ones for accountability and transparency;

b. Asset Protection. Assets of DLT systems (e.g. oracles, smart contracts, digital wallets) should be securely protected. DLT service providers should implement asset protection services through access control (ability to determine who is able to govern and change the state of an asset), application security, platform security, network security and physical security to provide logical and physical protection to assets residing in DLT systems;

c. Identity Management. The integrity of identities needs to be upheld throughout their life cycles, with the scope covering entities and roles in DLT

[9] Adapted from ISO 23257:2022 Blockchain and Distributed Ledger technologies (DLT) Reference Architecture

systems and non-DLT systems. Examples of identity management services, processes and tools include role management and logging;

d. Cryptography and Key Management. The management of cryptographic keys is essential for the identity management and confidentiality of DLT systems. DLT service providers should adopt DLT systems with cryptographic algorithms from well-established international standards. DLT service providers should also select DLT systems with appropriate algorithms and encryption key lengths that meet their security objectives and requirements based on existing standards[10]. Developers and DLT service providers should be kept up-to-date with quantum-enabled threats as well as new weaknesses in existing algorithms and be ready to replace their cryptographic schemes with resilient post-quantum cryptographic schemes[11];

   DLT service providers should consider the following for the key management [12]:

   i. Segregation of keys from other informational assets in the system;
   ii. Robust access control of cryptographic keys (e.g. the use of HSM to store and protect keys);
   iii. Access limitation of keys as prolonged usage of keys increases the risk of abuse, leakage and theft; and
   iv. Contingency plans for loss of keys (e.g. back up of keys).

e. Security Assessment and Testing Management. The threat models of DLT systems and software architectures and design should be reviewed on a regular basis, complementing existing audit and assurance functions and activities.

[10] **Secure Hash Standard (SHS), NIST & Digital Signature Standard, FIPS 186-4**
[11] **Source: Post-Quantum Cryptography, NIST**
[12] **Source: ISO/TR 23576:2020 Blockchain and distributed ledger technologies - Security management of digital asset custodians**

DLT service providers could engage competent independent third parties to perform code reviews periodically to improve code quality and detect issues such as the implementation of backdoors by developers.

DLT service providers could engage competent independent third parties to perform penetration testing and host bug bounty programmes for an in-depth evaluation of cybersecurity defences. Penetration testing should be based on threat models and conducted on live production environments, with proper safeguards put in place in anticipation of any potential implications. If testing on live production environments is not feasible due to operational concerns, penetration testing may be performed on test environments that closely mimic production environments, with penetration testing performed on live production environment only for known differences. The frequency of penetration testing should be implemented based on factors such as system criticality and the system's exposure to risks. For systems that are directly assessable from the Internet, service providers are expected to conduct penetration testing at least once annually or whenever these systems undergo major changes or updates[13].

DLT service providers should consider adversarial attack simulation exercises (otherwise known as red-teaming) to stress and enhance the systems' abilities to detect and respond to real-world attacks from sophisticated adversaries.

DLT service providers should perform other assessment and testing activities such as validation testing, negative testing[14], logging, monitoring, periodic scanning of systems along with the reporting of findings and plans for mitigative measures;

f. Protection of Personal Data. When storing or processing personal data on DLT systems, it is essential to protect personal data from unauthorised access, especially on permissionless DLT systems. Some examples of capabilities that could support personal data protection include access control, data encryption,

[13] Source: MAS' Technology Risk Management Guidelines, January 2021.

[14] Examples of negative testing that handle unwanted inputs and user behaviour are buffer overflows and code injection.

anonymisation, de-identification, destruction, logging and monitoring. The Personal Data Protection Act should also be referenced for the development of security controls to safeguard personal data and the considerations on how external parties should protect personal data supplied to DLT-enabled services[15];

g. Availability Management. Availability management is aimed at ensuring that the functions and services involved in DLT systems are available and meet requirements stated in contracts, service level agreements or other documentation. DLT service providers should assess their capabilities on availability management such as capacity, provisioning, reliability, maintainability, logging and monitoring against the requirements agreed upon by organisations using DLT-enabled services and/or other stakeholders. Service providers should also implement robust business continuity plans (BCP) for [16]greater resiliency, enhancing the ability to recover from and minimise the impact of cyber-attacks or operational failures; and

h. Access Management. Access management provides users and DLT systems the authentication and authorisation to log on and transact successfully. DLT systems should include services that implement processes to validate and/or verify the identity of users before they transact using DLT system or limit the actions users can undertake once successfully authenticated. Examples of access management features include identification, authentication, authorisation, access control and access logs.

[15] Source: Guide on Personal Data Protection Considerations for Blockchain Design, PDPC

[16]

6. Mitigating Known Attacks on Smart Contracts

The developer should design and validate the behaviour of the smart contract based on the owner's definition of business logic. This ensures that the smart contract does not deviate from the intended business logic. As smart contracts have shown to be error-prone in practice, it is vital to build a level of confidence in a smart contract and understand the limitations of the programming functions (e.g. transaction costs, time stamps and randomness). The guidance below pertains to components in the cross-layer functions of the DLT reference architecture, covering security-by-design and risk management.

Security-by-Design

Developers need to be aware of existing and new threats that their systems may be susceptible to and review the threat models for the implementation of new or additional control measures. There are four common threat modelling approaches to reflect possible known attacks to components or assets, with the goal of implementing counter measures against those threats (see Annex A for details of threat modelling approaches). Developers should have good knowledge of the system specifications to act promptly when new vulnerabilities applicable to their systems are made known.

Developers should adhere to security-by-design and zero trust principles [17] **in developing a smart contract** (i.e. thorough and updated testing, user authentication, authorisation and access control policies). By applying security-by-design principles to drive smart contract development and code re-use, this can minimise errors and vulnerabilities during the execution of smart contracts. Security-by-design principles can be manifested through the use of verified and trusted templates, static and dynamic code analysis, and formal verification. For smart contracts to adhere to zero trust

[17] Source: Examples of best practices include Ethereum Secure Development Guidelines, ISACA: Blockchain Framework and Guidance and EOS Secure Development Guidelines.

principles, smart contracts should be tested thoroughly, with new tests added whenever new attack vectors are discovered.

Smart contracts should reference trusted templates where appropriate, since development from scratch could bring higher risks. Developers are recommended to use templates from libraries that provide secure and verified codes [18][19]. This will allow developers to concentrate on components that have not been formally verified and audited yet.

Developers should identify vulnerabilities of smart contracts during system development phases which include designing, coding, building, and testing**.** This requires vulnerability assessment to be conducted earlier during system development phases, instead of at the end-of-systems development which could incur more resources to address vulnerabilities.

a. Static code analysis provides a wide range of pre-configured rules for validation. This enables a higher success rate of parsing and code analysis without crashing and it also has a lower false positive rate in identifying vulnerabilities;
b. Dynamic analysis helps to detect crashes and other failures in binaries and automatically generates minimum test-cases for quick triage;
c. Formal verification is conducted at the bytecode level and analyses the behaviour of the smart contract vis-à-vis the intended specification. This is primarily for untrusted or unverified smart contract code and serves as an additional layer of defence to developers when testing smart contracts; and
d. Manual auditing should be conducted by independent experts through source code reviews to identify potential bugs and vulnerabilities that are undetected by automated tools.

Service providers should engage independent parties to perform smart contract audits (e.g. secure and verified codes, formal verification, static and dynamic code analysis) to identify and mitigate smart contract security risks. Service providers need to conduct

[18] Source: Gartner – Designing Blockchain Smart Contract Security and Access Control
[19] Source: ETSI GS PDL 011 V1.1.1 Permissioned Distributed Ledger (PDL): Specification of Requirements for Smart Contracts' Architecture and Security

their own evaluation of such independent parties in terms of competency and track record. Independent audits should also be performed before such changes go live in the production environment.

Service providers of smart contracts should ensure the processes surrounding the smart contracts are properly documented. The documentation of smart contract comprises technical specifications, deployment status, procedures (e.g. responsible disclosure policies, recourse in case of failure, use of open-source code), known issues (e.g. known vulnerabilities, potential attacks), history (e.g. test reports), and contacts (e.g. parties to engage for issues).

Governance and Risk Management

Service providers should enforce governance frameworks and mechanisms to uphold security-by-design and have a systematic risk-based process to prioritise controls to secure smart contracts against and respond to known and emerging threats:

a. **Revisit security policies and frameworks** to assure that risks introduced by DLT systems are adequately addressed by organisations' security policies;
b. **Oversee the use of open-source codes** to mitigate the risks of using such code in DLT-enabled services. This would subject the use of open-source code to review, testing and oversight before they are integrated into the DLT-enabled services;
c. **Implement change management processes** to assure that appropriate testing and approvals are obtained prior to deployment to avoid exploitation of security vulnerabilities; and
d. **Implement incident management programmes** to ensure that incidents are contained in a timely manner.

Service providers should define the smart contract baseline behaviour and identify deviations that indicate malicious or anomalous behaviour. With the establishment of baseline behaviour of smart contracts, smart contracts are given boundaries on how they should interact with the components found within DLT systems. Smart contracts should be designed so that they can be paused or terminated when things go wrong (e.g. through circuit breakers).

If DLT service providers are using third party suppliers to develop and deploy smart contracts, they should validate evidence provided by third party suppliers on how information security and risk management have been incorporated in the development of smart contracts.

Service providers should continually monitor and manage risks through advanced analytics and real-time monitoring to identify, detect and respond to anomalous activities at the earliest opportunity. Such capabilities can raise risk management productivity, with the growing complexity of security and controls across business functions and the variety of smart contract use cases.

7. Mitigating Known Attacks on Digital Wallets

This section provides best practices for the developers and service providers to adopt and address key security threats for digital wallets.

Service providers should ensure that the wallets of organisations using DLT-enabled services are secured to prevent digital payment tokens from being stolen. This could be in the form of either (i) hosted wallets or (ii) self-custody wallets.

i. **For hosted wallets, service providers need to ensure that they adhere to the following requirements** [20]**:**
   a. The digital payment tokens are managed in cold (hardware) wallets and vaulted in highly secure locations;
   b. Network isolation between hosted wallet systems and other information systems to prevent unauthorized connections through information systems;
   c. Network isolation of hosted wallet systems from the Internet (e.g. through locating hosted wallet systems in the DMZ, restrictions of programmatic and interactive accesses by the hosted wallet systems); and
   d. Multiple employees and approvals are required to physically access the hardware wallets.

ii. **While hosted wallets is an option for organisations using DLT-enabled services to manage digital payment tokens**, another option is to use **self-custody wallets**. However, organisations using DLT-enabled services need to be aware that they are responsible for maintaining the security of their self-custody wallets.

[20] Adapted from ISO/TR 23576 Blockchain and distributed ledger technologies — Security management of digital asset custodians

Service providers could provide guidance to their organisations using DLT-enabled services on how their self-custody wallets and digital payment tokens can be secured against attacks and scams. With such guidance, organisations using self-custody wallets would be able to use cold (hardware) and hot (software) wallets to store private keys on physical devices with specialised firmware or in the desktops/browsers. Cold and hot wallets could serve as a form of multi-factor authentication when using multi-signatures[21]. Annex B provides details on the guidance that service providers can provide to organisations using self-custody wallets.

[21] Source: A Security Reference Architecture for Blockchains, Ivan Homoliak, Sarad Venugopalan, Qingze Hum and Pawel Szalachowski, 2019

8. Terms and Definitions

Terms	Definitions
architecture	fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution
assets	anything that has value to a stakeholder
block	structured data comprising a block header and block data
block data	structured data comprising zero or more transaction records or references to transaction records
block header	structured data that includes a cryptographic link to the previous block unless there is no previous block
blockchain	distributed ledger with confirmed blocks organised in an append-only, sequential chain using cryptographic links
blockchain system	system that implements a blockchain
bytecode	DLT instructions for execution by smart contract
circuit breakers	mechanism to stop execution if certain conditions are met and are useful when new errors are discovered
cold wallet	offline (hardware) mechanism for storing private and public keys which enables DLT and blockchain users to transact
confirmed	accepted by consensus for inclusion in a distributed ledger
consensus	agreement among DLT nodes that (1) a transaction is validated and (2) that the distributed ledger contains a consistent set and ordering of validated transactions
consensus mechanism	rules and procedures by which consensus is reached
cryptocurrency	digital payment token designed to work as a medium of value exchange
cryptography	discipline that embodies the principles, means, and methods for the transformation of data in order to hide their semantic content, prevent their unauthorized use, or prevent their undetected modification
decentralised application (DApp)	application that runs on a decentralised system
decentralised finance (DeFi)	decentralised financial transactions and services that are accessible to anyone who can use the DLT system
developer	party that develops DLT and applications for service providers
digital asset	asset that exists only in digital form or which is the digital representation of another asset
digital payment token	any cryptographically-secured representation of value that is used or intended to be used as a medium of exchange

digital wallet	application used to generate, manage, store, or use private and public keys. A digital wallet can be implemented as a hardware or software module.
Distributed ledger	ledger that is shared across a set of DLT nodes and synchronized between the DLT nodes using a consensus mechanism
distributed ledger technology (DLT)	a technological approach that keeps records of transactions (ledger) which shared across a set of distributed networks of computer server (nodes) and synchronised among DLT nodes using a consensus mechanism
DLT network	network of DLT nodes which make up a DLT system
DLT node	device or process that participates in a network and stores a complete or partial replica of the ledger records
DLT platform	set of processing, storage and communication entities which together provide the capabilities of the DLT system on each DLT node
DLT system	system that implements a distributed ledger
DLT service providers	third-party companies offering DLT-based platforms, infrastructure and applications under service agreements
formal verification	analysis of smart contract's behaviour vis-à-vis the intended specification at the bytecode level using formal mathematical methods
functional component	functional building block needed to engage in an activity, backed by an implementation
hot wallet	online (software) mechanism for storing private and public keys which enables DLT and blockchain users to transact
hosted wallets	centralised platforms providing interfaces for interaction with wallets
immutability	property wherein ledger records cannot be modified or removed once added to a distributed ledger
ledger	information store that keeps records of transactions that are intended to be final, definitive, and immutable
multi-signature wallet	a signature scheme which requires multiple isolated signing keys to sign a message for it to be valid. It is an effective method of protecting each key, which is held by different entities, as each key cannot be used on its own
negative testing	a method of testing an application or system that ensures the plot of the application is according to the requirements and can handle unwanted input and user behaviour
non-DLT system	system outside a DLT system that a DLT system communicates with in order to accomplish its business goals
off-chain	related to a DLT system, but located, performed, or run outside that DLT system
on-chain	located, performed, or run inside a DLT system
oracles	represent trusted systems designed to supply external data to a DLT system or to respond to events from DLT systems

peer-to-peer	relating to, using, or being a network of equal peers that share information and resources with each other directly without relying on a central entity
penetration testing	use of a variety of manual and automated techniques to simulate attacks on DLT and applications by qualified and independent ethical security testers
permissioned DLT system	DLT system in which authorisation is required to perform any activity
permissionless DLT system	DLT system that does not require authorisation to perform any particular activity
personally identifiable information	information that (a) can be used to establish a link between the information and the natural person to whom such information relates, or (b) is or can be directly or indirectly linked to a natural person
private key	key of an entity's asymmetric key pair that is kept secret and which should only be used by that entity
public testnet	an alternative blockchain developers use to test application in a near-live environment. Tests are run on the testnet to ensure that the protocol is working as intended and avoiding users to pay gas fees when deploying smart contracts
red-teaming	adversarial attack simulation exercises that stress and enhance DLT systems' abilities to detect and respond to real-world attacks from sophisticated adversaries
secret sharing	dividing the signature key into multiple parts, then managing them with multiple isolated systems. It is an effective measure to protect the keys against leakage and theft
smart contract	computer program stored in a DLT system wherein the outcome of any execution of the program is recorded on the distributed ledger
threat model	structured representation of all the information that affects the security of an application
threshold signature	a cryptographic primitive for the generation, management and use of distributed keys, which leverages multi-party computation concepts.
Token	digital asset that represents a collection of entitlements
transaction	smallest unit of a work process, which is one or more sequences of actions required to produce an outcome that complies with governing rules
validation	function by which a transaction, ledger record, or block is validated
validation testing	a method of testing an application or system that ensures the product complies with the system requirements and performs the dedicated functions designed
wallet	application used to generate, manage, store, or use private and public keys

zero trust	premise that trust is never granted implicitly but must be continually evaluated

9. Abbreviations and Acronyms

Acronym	Meaning
Dapp	Decentralised Applications
DeFi	Decentralised Finance
DLT	Distributed Ledger Technology
DPT	Digital Payment Tokens
NFT	Non-fungible Tokens
PII	Personally Identifiable Information
HSM	Hardware Security Module

10. Analysis of Major DLT Security Events from 2011 to June 2022

Victim	Amount stolen (USD)	Service Type	Hack Type	Exploit	Corresponding DLT component
Harmony Bridge	$100 million	Defi Platform	Security Breach	Stolen private keys	Cryptography & Key Management
Ronin Network (Axie Infinity)	$650 million	DeFi Platform	Security Breach	Stolen private keys	Cryptography & Key Management
Poly Network	$610 million	DeFi Platform	DeFi Breach (Code Exploit)	Bug exploited in smart contract	Smart Contract
Coincheck	$532 million	Exchange	Security Breach	Hot wallet compromised	Digital Wallets
MT Gox	$470 million	Exchange	Security Breach	Stolen private keys	Digital Wallets
Wormhole	$326 million	DeFi Platform	DeFi Breach (Code Exploit)	Bug exploited in smart contract	Smart Contract
KuCoin	$281 million	Exchange	Security Breach	Hot wallet compromised	Digital Wallets
PancakeBunny	$200 million	DeFi Platform	DeFi Breach (Flash Loan)	Bug exploited in smart contract	Smart Contract
Bitmart	$196 million	Exchange	Security Breach	Hot wallet compromised	Digital Wallets
Beanstalk	$182 million	DeFi Platform	DeFi Breach (Flash Loan)	Bug exploited in smart contract	Smart Contract
Bitgrail	$150 million	Exchange	Security Breach	Cold wallets compromised	Cryptography & Key Management
BadgerDAO	$150 million	DeFi platform	Security Breach	Phishing	Cryptography & Key Management
Venus	$150 million	DeFi platform	DeFi Breach (Code Exploit)	Bug exploited in smart contract	Smart Contract

Victim	Amount stolen (USD)	Service Type	Hack Type	Exploit	Corresponding DLT component
Cream Finance	$130 million	DeFi platform	DeFi Breach (Flash Loan)	Bug exploited in smart contract	Smart Contract
CoinBene	$105 million	Exchange	Security Breach	Hot wallet compromised	Digital Wallets
Vulcan Forged	$103 million	Gaming Platform	Security Breach	Stolen private keys	Cryptography & Key Management
Liquid	$97 million	Exchange	Security Breach	Hot wallet compromised	Digital Wallets
DAO Hack	$70 million	DeFi platform	DeFi Breach (Code Exploit)	Bug exploited in smart contract	Smart Contract
Parity Wallet	$ 30 million	Wallet Provider	Security Breach	Bug exploited in multi-signature wallet code	Smart Contract

Table 2. Analysis of major DLT incidents and attacks from 2011 to June 2022

11. Analysis of Singapore-Based DLT Events from 2018 to June 2022

Victim	Amount stolen (USD)	Service Type	Hack Type	Exploit	Corresponding DLT component
Kickico.co	$7.7 million	Exchange	Security Breach	Stolen private keys	Cryptography & Key Management
BitTrue	$5 million	Exchange	Security Breach	Hot wallet compromised	Digital Wallets
CoinTiger	$1.8 million	Exchange	Security Breach	Cold wallet compromised	Digital Wallets
CoinHako	Undisclosed	Exchange	Security Breach	Undisclosed	Nil
KuCoin	$281 million	Exchange	Security Breach	Hot wallet compromised	Digital Wallets
Vulcan Forged	$103 million	Gaming Platform	Security Breach	Stolen private Keys	Cryptography & Key Management
Crypto.com	$33 million	Exchange	Security Breach	Undisclosed	Nil
Ronin Network (Axie Infinity)	$650 million	DeFi Platform	Security Breach	Stolen private keys	Cryptography & Key Management

Table 3. Analysis of Singapore-based DLT events from 2018 – June 2022

12. References

S/N	Document	Source	Dated
1	Federal Information Processing Standard (FIPS) 186-4, Digital Signature Standard	NIST	2015
2	Distribute Ledger Technology Terms and Definition	ITU-T	2019
3	EOS Secure Development Guidelines	Slowmist	2019
4	The Security Reference Architecture for Blockchains	SUTD	2019
5	ISO/TR 23576 Blockchain and distributed ledger technologies — Security management of digital asset custodians	ISO	2020
6	ISO 22739 Blockchain and distributed ledger technologies — Vocabulary	ISO	2020
7	Post-Quantum Cryptography	NIST	2020
8	ISACA: Blockchain Framework and Guidance	ISACA	2020
9	Crypto & DeFi Hacks & Scams Report	Crystal Blockchain	2021
10	Designing Blockchain Smart Contract Security and Access Control	Gartner	2021
11	Design Considerations for Blockchain Smart Contracts	HCL Technologies	2021
12	ETSI GS PDL 011 V1.1.1 Permissioned Distributed Ledger (PDL): Specification of Requirements for Smart Contracts' Architecture and Security	ETSI	2021
13	Blockchain Networks: Token Design and Management Overview	NIST	2021
14	TR 68-3: Technical Reference for Autonomous vehicles -part 3: Cybersecurity principles and assessment framework	Singapore Standards	2021
15	ISO 23257:2022 Blockchain and Distributed Ledger technologies (DLT) Reference Architecture	ISO	2022
16	Ethereum Secure Development Guidelines	Consensys	2022
17	Guide on Personal Data Protection Considerations for Blockchain	PDPC	2022

Annex A – Secure Development of Smart Contracts with Security-By-Design Principles

Background

Smart contract is a contract written in DLT-specific programming language that can automate contracts or agreements for decentralised applications. Smart contracts are used to hold funds and store public DLT states under the contract's address. The smart contracts in the DLT system can be called upon and deployed for other contracts [22][23].

Secure Development of Smart Contracts

The approach of developing secure smart contracts is as follows [24][25]:

1. Defining smart contract behaviour. Developers should define and determine the terms of agreement within its scope of participating stakeholders. Next, developers should identify involved parties in the execution of smart contracts and determine the approach for consensus. Developers should also include flexibility for future code upgrading or changes so that the architecture is extensible to encompass new use cases:
    a. Determination of new terms for future versions;
    b. Development of new smart contracts to address vulnerabilities in the existing ones; and
    c. Renewal of smart contracts upon expiry.
2. Designing the smart contract. Developers should start by determining the events which could trigger the implementation of the contract. Developers should specify the data elements, assess if there are any inputs that impact the

[22] Source: ISO 23257:2022 Blockchain and Distributed Ledger technologies (DLT) Reference Architecture
[23] Source: ETSI GS PDL 011 V1.1.1 Permissioned Distributed Ledger (PDL): Specification of Requirements for Smart Contracts' Architecture and Security
[24] Source: Design Considerations for Blockchain Smart Contracts - HCL Technologies
[25] Source: Ethereum Secure Development Guidelines - Consensys

DLT's execution and identify the limitations of the underlying DLT system. In addition, the approach on how the parties involved ought to interact through the smart contract should be clearly stated;

3. Coding the business logic. Developers can program the conditions of execution as specified in the business logic;
4. Testing the smart contract. The developed set of codes should be tested thoroughly in a public testnet, with tests added when new attack vectors are added. Smart contracts should be deployed in phases, with increasing usage and testing in each phase;
5. Executing the smart contract. With the execution of the contract, the output from the transaction is stored on the DLT; and
6. Updating the DLT. When the ledger nodes are updated with changes in DLT states, the new updates are reflected in the DLT system.

Security-By-Design Principles for Smart Contracts

Smart contracts are developed using various programming languages such as Solidity, Vyper or Java, etc. It is essential to adhere to best practices for developing secure smart contracts. Below are key design principles for developers to create smart contracts on DLT platforms [26][27]:

a. Keep the smart contract code clear and understandable. Developers should ensure that the contract addresses a specific problem to minimise design and coding errors. Developers should also modularise the code to keep contracts and their underlying functions manageable. Developers should only use the DLT for the parts of the system that require decentralisation;
b. Decide on the required data to be kept on the smart contract. Developers should examine the scope of application data and determine what should be kept on-chain and off-chain. More specifically, developers should state variables for the smart contract to be stored efficiently on-chain data and leave data stored off-chain to be managed by external systems;

[26] Source: Design Considerations for Blockchain Smart Contracts – HCL Technologies
[27] Source: Ethereum Secure Development Guidelines - Consensys

c. Embed processes to address security failures. Developers should ensure that smart contract code can handle errors and ensure that the smart contract can be paused or terminated (e.g. through circuit breakers) when a deviation is detected or when there is a need for upgrades and bug fixes. Developers can consider including speed bumps to slow down actions so that there is enough time to recover when malicious actions occur. Developers should validate that the smart contract has a limit on the amount of digital payment tokens that it can handle within a time interval (e.g. rate limiting), thus limiting the impact of breaches of the smart contract;
d. Protect data. Hashing or encryption protects the confidentiality of data in a DLT platform since data in the DLT is viewable by the public. Developers should ensure that a secure hash or encryption should be used to protect the visibility of DLT data;
e. Define access controls. Access modifiers should be used to define the accessibility rules of the smart contract. It ensures that only appropriate parties have the authorisation to perform critical functions that affect the smart contract; and
f. Specify conditions and/or a period of validity for smart contracts, following which they will expire and be deactivated.

Threat Modelling for Smart Contracts

Developers need to be aware of existing and new threats that their systems may be susceptible to and review the threat models for the implementation of new or additional control measures. Developers should have good knowledge of the specifications of the DLT-enabled services to act promptly when new vulnerabilities applicable to their DLT-enabled services are made known [28].

There are four common threat modelling approaches to reflect possible known attacks to components or assets with the goal of implementing countermeasures against those threats:

[28] Source: TR 68-3: Technical Reference for Autonomous vehicles -part 3: Cybersecurity principles and assessment framework

a. Software-centric approach. This approach involves designing systems using illustrative software architecture diagrams such as data flow, use case, and component. STRIDE is one such example. It uses the identification of system entities, events, and the boundaries of the system to accomplish what it does;
b. Asset-centric approach. This approach involves identifying the assets of an organisation that have been entrusted to a system or has software-related data classified according to their intrinsic value and is attractive to a potential attack. This facet can be used for prioritising risk. PASTA is an example that involves threat modelling using a seven-step process of attack simulation and threat analysis (PASTA);
c. Attacker-centric approach. This approach requires building up an attacker's profile based on skillset and the motivation to exploit vulnerabilities and system characteristics. For example, an attack could be carried out along any point of a tree diagram mapping both software and assets to help envisage various attack patterns. Attack trees can be used either as a standalone or be combined with other threat modelling approaches such as PASTA. STRIDE usually starts with an attack goal at the tree root and breaks into multiple tree branches; and
d. Threat-centric approach. This approach recognises that attackers need a point-of-origin to initiate their attack, i.e. an "attack surface" using a range of means to introduce an unknown threat agent that could lead to single or multiple attack paths (including targeted assets). For example, an organisation building its own integrated threat protection might want to deploy an enhanced threat detection system that proactively scans its environment. Detected threats can then be remediated based on the risk likelihood and impact.

Annex B – Secure Provisioning of Digital Wallets

Background

DPT is any cryptographically-secured representation of value that is used or intended to be used as a medium of exchange on DLT. In addition, they often rely on smart contracts and are used with decentralised applications. These tokens can be exchanged and represented as digital assets.

In DLT, digital payment tokens are managed through two different types of wallets:

a. Self-custody wallets. These are hot and cold wallets where private keys are locally stored. The keys are also directly interacting with the DLT platform for the signing of transactions; and
b. Hosted wallets. Centralised platforms provide interfaces for interaction with wallets. The private keys are fully controlled by third parties [29]

Limitations of Digital Wallets

Hosted wallets can be subjected to availability attacks such as DoS and jamming. While self-custody wallets store the private keys locally and do not reveal them to a centralised party, organisations using DLT-enabled services must trust the online interface provided by the third party with the availability of this interface is dependent on this party [30]. An example of a security threat to self-custody wallets is phishing attacks, which could divert users to fake websites.

Securing Digital Wallets

While hosted wallets is an option for organisations using DLT-enabled services to manage digital payment tokens, another option is to use self-custody wallets. However,

[29] Source: A Security Reference Architecture for Blockchains, Ivan Homoliak, Sarad Venugopalan, Qingze Hum and Pawel Szalachowski, 2019
[30] Source: A Security Reference Architecture for Blockchains, Ivan Homoliak, Sarad Venugopalan, Qingze Hum and Pawel Szalachowski, 2019

organisations using DLT-enabled services need to be aware that they are responsible for maintaining the security of their self-custody wallets. If organisations using DLT-enabled services prefer to use their own digital wallets, they should consider using hot and cold wallets as co-signing wallets. Specific recommendations relate to securing digital wallets using multi-signatures:

a. Transfer digital payment tokens from hosted wallets to cold wallets;
b. Choose a multi-signature cold wallet;
    i. Organisations using DLT-enabled services should use two different multi-signatures wallets for managing their digital assets and digital payment tokens.
c. Implement multi-signatures that serve as multi-factor authentication to increase the difficulty of attackers stealing funds from the wallet and can be used as key recovery:
    i. Organisations using DLT-enabled services should pick *an m-of-n* multi-signature. Where m refers to the number of signatures or authorisation and n refers to the number of Co-signers. An example of an m-of-n option is 2-of-3, which requires signatures from at least two wallets for the transaction to be conducted.
        1. First Co-signer – Software wallet (Desktop);
        2. Second Co-signer – Software/Hardware wallet (Offline); and
        3. Third Co-signer – Software/Hardware wallet (Offline).
    ii. Storing multiple keys in different wallets can function as a backup for business continuity and security purposes [31].

In addition, DLT service providers should advise organisations using DLT-enabled services to adopt the following precautions, with the mindset that they are responsible for the security of their digital wallets and tokens should they choose the option of self-custody wallets:

a. Never share secret phrases of private keys with others;
b. Never connect digital wallets to untrusted sites;

[31] **Source: ISO TR 23576:2020 Blockchain and distributed ledger technologies — Security management of digital asset custodians**

c. Back up your private keys in the event of accidental loss of private keys;
d. Do not interact with unknown parties and trust deals that are too good to be true (e.g. winning of digital payment tokens from unknown sources, purchase of NFTs from unverified origins); and
e. Do not download wallet applications from unverified origins.

QUERIES & FEEDBACK

Questions and feedback on this document may be submitted to:

Contact@csa.gov.sg