CompileX

Litepaper v0.4

Bitcoin-inspiriertes digitales Asset für Builder und Developer

Stand: 31.07.2026 | Netzwerk: Ethereum Sepolia | Symbol: CMPX

Feld	Wert
Projekt	CompileX
Token	CMPX
Website	https://compilex-token.dev
GitHub	https://github.com/CompileX-Token/compilex-cmpx-token
Release	https://github.com/CompileX-Token/compilex-cmpx-token/releases/tag/v0.1.0
Contract	0xa1A09ba7357c5187d90E1De160bFd868151f3D8C

Wichtiger Hinweis
CompileX ist aktuell ein Sepolia-Testnet-Prototyp. Dieses Dokument ist kein Verkaufsangebot, keine Anlageberatung und kein Gewinnversprechen.

1. Kurzfassung

CompileX ist ein Bitcoin-inspiriertes digitales Asset für Entwickler, Builder und technische Communities. Der aktuelle CMPX-Prototyp ist ein verifizierter ERC-20-Token auf dem Ethereum-Sepolia-Testnet.

Der Fokus liegt auf digitaler Knappheit, transparenter Tokenlogik und praktischer Nutzbarkeit. Seit der vorherigen Version wurden die öffentliche GitHub-Struktur, der erste GitHub Release, die Roadmap-Seite und eine Checkout Demo ergänzt.

CompileX ist aktuell ein experimentelles Testnet-Projekt. Dieses Litepaper stellt kein öffentliches Verkaufsangebot, keine Anlageberatung und kein Gewinnversprechen dar.

2. Aktueller Projektstatus

Der Stand von CompileX v0.4 ist:

- Sepolia-Contract deployed und verifiziert
- Smart Contract öffentlich auf Etherscan einsehbar
- GitHub Repository öffentlich
- GitHub Actions Workflow erfolgreich eingerichtet
- erster öffentlicher Release veröffentlicht
- Landingpage, Rechtseiten und Roadmap online
- CompileX Pay Demo und Checkout Demo online
- Litepaper v0.4 als aktueller Dokumentationsstand

Der aktuelle öffentliche Release ist CompileX v0.1 - Sepolia Testnet Prototype.

3. Designprinzipien

CompileX orientiert sich an einfachen, überprüfbaren Regeln:

- feste maximale Tokenmenge
- keine nachträgliche Mint-Funktion
- keine Owner- oder Admin-Rollen
- keine Blacklist
- keine Transfersteuer
- Burn-Funktion aktiv
- Permit/EIP-2612-Unterstützung aktiv
- öffentliche Contract-Verifizierung
- Fokus auf Builder, Developer-Tools und technische Anwendungen

CompileX soll nicht als „neues Bitcoin“ oder „Bitcoin 2.0“ verstanden werden. Die Formulierung „Bitcoin-inspiriert“ bezieht sich auf Knappheit, klare Regeln und überprüfbare Tokenlogik.

4. Token-Parameter

Eigenschaft	Wert
Name	CompileX
Symbol	CMPX
Max Supply	21.000.000 CMPX
Decimals	8
Minting	Nicht vorhanden
Owner/Admin	Nicht vorhanden
Burn	Aktiv
Permit	Aktiv
Aktuelles Netzwerk	Ethereum Sepolia
Chain ID	11155111
Contract	0xa1A09ba7357c5187d90E1De160bFd868151f3D8C

5. Tokenomics v0.2

Die maximale Gesamtmenge beträgt 21.000.000 CMPX. Für die weitere Projektplanung wird folgende Verteilung als Arbeitsmodell verwendet:

Bereich	Anteil	Menge
Community / Rewards	40 %	8.400.000 CMPX
Ecosystem / Development	25 %	5.250.000 CMPX
Treasury	20 %	4.200.000 CMPX
Liquidity Reserve	10 %	2.100.000 CMPX
Early Supporter / Airdrop	5 %	1.050.000 CMPX

Diese Verteilung ist für die Testnet- und Konzeptphase vorgesehen. Vor einem möglichen Mainnet-Start müssen technische, rechtliche, steuerliche und organisatorische Fragen geprüft werden.

6. CompileX Pay Demo

CompileX Pay ist eine Testnet-Demo für einfache CMPX-Zahlungen. Die Demo ist unter /pay.html erreichbar und nutzt den verifizierten CMPX-Contract auf Sepolia.

6.1 Ziele der Demo

Die Pay Demo soll zeigen, dass CMPX technisch für einfache Zahlungen nutzbar ist:

- Wallet verbinden

- CMPX zu MetaMask hinzufügen
- Empfängeradresse eintragen
- CMPX-Betrag festlegen
- Payment Request erzeugen
- CMPX über MetaMask senden
- Transaktionsstatus prüfen
- Empfänger-Balance abfragen
- Transaktion auf Sepolia Etherscan öffnen

6.2 Voraussetzungen

Der sendende Account benötigt CMPX für den Token-Transfer und SepoliaETH für die Netzwerkgebühr. Auch bei einem CMPX-Transfer wird Gas in SepoliaETH bezahlt. Wenn der sendende Account kein SepoliaETH besitzt, kann MetaMask die Transaktion nicht ausführen.

6.3 Technischer Ablauf

Die Pay Demo erzeugt eine ERC-20-Transfer-Interaktion mit dem CMPX-Contract. Beim Klick auf „CMPX senden“ wird MetaMask geöffnet und eine Transaktion an den CMPX-Contract vorbereitet. Die Transaktion nutzt die transfer(address,uint256)-Funktion des Tokens.

Nach dem Senden wartet die Website auf eine Transaktionsbestätigung und zeigt anschließend den Status an. Zusätzlich kann die Empfänger-Balance per balanceOf(address) geprüft werden.

7. CompileX Checkout Demo

Die Checkout Demo ergänzt die Pay Demo um einen greifbaren Beispiel-Use-Case: den Developer Access Pass.

Die Demo zeigt eine einfache Testnet-Kasse mit:

- Beispielprodukt: Developer Access Pass
- Preis: 25 CMPX
- Demo-Händleradresse
- Zahlungsbutton über MetaMask
- Statusanzeige offen / wartet / bezahlt
- Transaktionslink zu Sepolia Etherscan
- Kopierfunktion für Contract- und Händleradresse

Die Checkout Demo ist kein echter Verkauf und schaltet keinen realen Zugang frei. Sie zeigt nur, wie CMPX als technischer Zahlungs- oder Zugangstoken in einer späteren Anwendung verwendet werden könnte.

8. Website, GitHub und Release-Stand

Die CompileX-Website enthält aktuell Landingpage, Tokenomics, Litepaper v0.4, Rechtseiten, CompileX Pay, Pay Guide, Checkout Demo und Roadmap-Statusseite.

Das öffentliche GitHub Repository enthält den Smart Contract, Tests, Deployment-Dokumentation, Tokenomics-Dokumentation, Pay-Demo-Dokumentation und Sicherheitsnotizen.

Der erste öffentliche GitHub Release dokumentiert den Stand CompileX v0.1 - Sepolia Testnet Prototype.

Ressource	Link
Website	https://compilex-token.dev
GitHub Repository	https://github.com/CompileX-Token/compilex-cmpx-token
Release v0.1	https://github.com/CompileX-Token/compilex-cmpx-token/releases/tag/v0.1.0
Sepolia Token	https://sepolia.etherscan.io/token/0xa1A09ba7357c5187d90E1De160bFd868151f3D8C

Ressource	Link
Verified Contract Code	https://sepolia.etherscan.io/address/ 0xa1A09ba7357c5187d90E1De160bFd868151f3D8C#code

9. Sicherheit und Mainnet-Readiness

CompileX ist aktuell bewusst ein Testnet-Projekt. Ein Mainnet-Start sollte erst nach technischer, rechtlicher und organisatorischer Vorbereitung stattfinden.

Vor einem möglichen Mainnet-Deployment sind mindestens diese Punkte offen:

- erweiterte Tests
- statische Analyse
- externer Smart-Contract-Review
- Prüfung der Tokenomics und Wallet-Struktur
- Treasury- und Multisig-Konzept
- rechtliche und steuerliche Prüfung
- finale Risikokommunikation
- finaler Launch-Plan

Private Keys, Seed-Phrases und .env-Dateien dürfen niemals veröffentlicht oder geteilt werden.

10. Roadmap

Phase	Beschreibung	Status
Phase 1 - Prototyp	Contract, Tests, Sepolia-Deployment und Contract-Verifizierung	abgeschlossen
Phase 2 - Website und Dokumentation	Landingpage, Litepaper, Tokenomics, Rechtseiten und technische Dokumentation	abgeschlossen
Phase 3 - Pay und Checkout	Testnet-Zahlungen mit CMPX, Statusprüfung, Pay Guide und Checkout Demo	abgeschlossen
Phase 4 - GitHub Release	Öffentliches GitHub Repository, GitHub Actions, Roadmap-Seite und Release v0.1	abgeschlossen
Phase 5 - Security Review Vorbereitung	Erweiterte Tests, Security-Checklist, Review-Dokumentation und externe Rückmeldung	nächster Schritt
Phase 6 - Developer Utility Prototyp	Weitere Beispiel-Use-Cases wie API-Credits, Tool-Zugang, Templates oder Community-Rewards	geplant
Phase 7 - Mainnet-Entscheidung	Erst nach Audit, rechtlicher Prüfung, klarer Projektstruktur und finaler Dokumentation	nicht gestartet

11. Risiken und Hinweise

CompileX befindet sich in einer frühen experimentellen Phase. Testnet-Token haben keinen zugesicherten wirtschaftlichen Wert. Es gibt keine Gewinnversprechen, keine Anlageberatung und kein öffentliches Verkaufsangebot.

Blockchain-Interaktionen können technische Risiken enthalten. Nutzer sind selbst für ihre Wallets, privaten Schlüssel und Seed-Phrases verantwortlich. Private Schlüssel oder Seed-Phrases werden von CompileX niemals benötigt.

Die Pay Demo und Checkout Demo verwenden Testnet-Transaktionen. Sie sind keine Zahlungsforderung, kein Shop-System mit realer Ware und keine rechtliche Abrechnung.

12. Nächste empfohlene Schritte

1. Litepaper v0.4 auf Website und GitHub veröffentlichen.
2. GitHub Release-Notizen mit dem neuen Dokumentationsstand verknüpfen.
3. Security-Review-Vorbereitung starten.
4. Tests und Dokumentation weiter ausbauen.
5. Developer-Utility-Prototyp definieren.
6. Mainnet-Readiness erst nach Prüfung weiter planen.