

CERTIK AUDIT REPORT FOR SWIPE



Request Date: 2019-08-15
Revision Date: 2019-08-19
Platform Name: Ethereum



px

Contents

Disclaimer	1
About CertiK	2
Exective Summary	3
Vulnerability Classification	3
Testing Summary	4
Audit Score	4
Type of Issues	4
Vulnerability Details	5
Manual Review Notes	6
Static Analysis Results	7
Formal Verification Results	8
How to read	8
Source Code with CertiK Labels	19

Disclaimer

This Report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Verification Services Agreement between CertiK and Swipe(the “Company”), or the scope of services/verification, and terms and conditions provided to the Company in connection with the verification (collectively, the “Agreement”). This Report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This Report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK’s prior written consent.

About CertiK

CertiK is a technology-led blockchain security company founded by Computer Science professors from Yale University and Columbia University built to prove the security and correctness of smart contracts and blockchain protocols.

CertiK, in partnership with grants from IBM and the Ethereum Foundation, has developed a proprietary Formal Verification technology to apply rigorous and complete mathematical reasoning against code. This process ensures algorithms, protocols, and business functionalities are secured and working as intended across all platforms.

CertiK differs from traditional testing approaches by employing Formal Verification to mathematically prove blockchain ecosystem and smart contracts are hacker-resistant and bug-free. CertiK uses this industry-leading technology together with standardized test suites, static analysis, and expert manual review to create a full-stack solution for our partners across the blockchain world to secure 6.2B in assets.

For more information: <https://certik.org/>

Executive Summary

This report has been prepared as the product of the Smart Contract Audit request by Swipe. This audit was conducted to discover issues and vulnerabilities in the source code of Swipe's Smart Contracts. Utilizing CertiK's Formal Verification Platform, Static Analysis, and Manual Review, a comprehensive examination has been performed. The auditing process pays special attention to the following considerations.

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessment of the codebase for best practice and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line by line manual review of the entire codebase by industry experts.

Vulnerability Classification

For every issue found, CertiK categorizes them into 3 buckets based on its risk level:

Critical

The code implementation does not match the specification, or it could result in loss of funds for contract owner or users.

Medium

The code implementation does not match the specification at certain conditions, or it could affect the security standard by lost of access control.

Low

The code implementation is not a best practice, or use a suboptimal design pattern, which may lead to security vulnerabilities, but no concern found yet.

Testing Summary

PASS

CERTIK believes this
smart contract passes security
qualifications to be listed on
digital asset exchanges.

Aug 19, 2019



Type of Issues

CertiK smart label engine applied 100% covered formal verification labels on the source code, and scanned the code using our proprietary static analysis and formal verification engine to detect the follow type of issues.

Title	Description	Issues	SWC ID
Integer Overflow and Underflow	An overflow/underflow happens when an arithmetic operation reaches the maximum or minimum size of a type.	0	SWC-101
Function incorrectness	Function implementation does not meet the specification, leading to intentional or unintentional vulnerabilities.	0	
Buffer Overflow	An attacker is able to write to arbitrary storage locations of a contract if array of out bound happens	0	SWC-124
Reentrancy	A malicious contract can call back into the calling contract before the first invocation of the function is finished.	0	SWC-107
Transaction Order Dependence	A race condition vulnerability occurs when code depends on the order of the transactions submitted to it.	0	SWC-114
Timestamp Dependence	Timestamp can be influenced by minors to some degree.	0	SWC-116
Insecure Compiler Version	Using an fixed outdated compiler version or floating pragma can be problematic, if there are publicly disclosed bugs and issues that affect the current compiler version used.	1	SWC-102 SWC-103
Insecure Randomness	Block attributes are insecure to generate random numbers, as they can be influenced by minors to some degree.	0	SWC-120

“tx.origin” for authorization	tx.origin should not be used for authorization. Use msg.sender instead.	0	SWC-115
Delegatecall to Untrusted Callee	Calling into untrusted contracts is very dangerous, the target and arguments provided must be sanitized.	0	SWC-112
State Variable Default Visibility	Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.	0	SWC-108
Function Default Visibility	Functions are public by default. A malicious user is able to make unauthorized or unintended state changes if a developer forgot to set the visibility.	0	SWC-100
Uninitialized variables	Uninitialized local storage variables can point to other unexpected storage variables in the contract.	0	SWC-109
Assertion Failure	The assert() function is meant to assert invariants. Properly functioning code should never reach a failing assert statement.	0	SWC-110
Deprecated Solidity Features	Several functions and operators in Solidity are deprecated and should not be used as best practice.	0	SWC-111
Unused variables	Unused variables reduce code quality	0	

Vulnerability Details

Critical

No issue found.

Medium

No issue found.

Low

No issue found.

Manual Review Notes

Source Code SHA-256 Checksum

- **SwipeWallet.sol**
c6275e908f1c61804c2ef4acb51ab1886dfe113640a785b815617148cdc506f8

Summary

CertiK was chosen by Swipe to audit the design and implementation of its SXP smart contract. To ensure comprehensive protection, the source code has been analyzed by the proprietary CertiK formal verification engine and manually reviewed by our smart contract experts and engineers. That end-to-end process ensures proof of stability as well as a hands-on, engineering-focused process to close potential loopholes and recommend design changes in accordance with the best practices in the space.

Overall we found the smart contracts to follow good practices. With the final update of source code and delivery of the audit report, we conclude that the contract is structurally sound and not vulnerable to any classically known anti-patterns or security issues. The audit report itself is not necessarily a guarantee of correctness or trustworthiness, and we always recommend to seek multiple opinions, keep improving the codebase, and more test coverage and sandbox deployments before the mainnet release.

Recommendations

Items in this section are low impact to the overall aspects of the smart contracts, thus will let client to decide whether to have those reflected in the final deployed version of source codes.

SwipeToken

- **INFO** `transfer()`, `transferFrom()`, `approve()`: Recommend checking to `!= address(0)`.
- **INFO** `transfer()`, `transferFrom()`: Recommend adding balance check and providing corresponding error messages, thus being consistent with `burn()`, `burnForAllowance()`.

Static Analysis Results

INSECURE_COMPILER_VERSION

Line 1 in File SwipeWallet.sol

```
1 pragma solidity ^0.5.0;
```



 Only these compiler versions are safe to compile your code: 0.5.10

Formal Verification Results

How to read

Detail for Request 1

transferFrom to same address


Verification date	 20, Oct 2018
Verification timespan	 395.38 ms

CERTIK label location	Line 30-34 in File howtoread.sol
-----------------------	----------------------------------

CERTIK label	30	/*@CTK FAIL "transferFrom to same address"
	31	@tag assume_completion
	32	@pre from == to
	33	@post __post.allowed[from][msg.sender] ==
	34	*/

Raw code location	Line 35-41 in File howtoread.sol
-------------------	----------------------------------


Raw code	35	function transferFrom(address from, address to
) {
	36	balances[from] = balances[from].sub(tokens
	37	allowed[from][msg.sender] = allowed[from][
	38	balances[to] = balances[to].add(tokens);
	39	emit Transfer(from, to, tokens);
	40	return true;
	41	}

Counterexample	 This code violates the specification	
Initial environment	1	Counter Example:
	2	Before Execution:
	3	Input = {
	4	from = 0x0
	5	to = 0x0
	6	tokens = 0x6c
	7	}
	8	This = 0
	52	}
	53	balance: 0x0
	54	}
	55	}
Post environment	57	After Execution:
	58	Input = {
	59	from = 0x0
	60	to = 0x0
	61	tokens = 0x6c

Formal Verification Request 1

SafeMath add

 19, Aug 2019

 33.88 ms

Line 36-41 in File SwipeWallet.sol

```
36      /*@CTK "SafeMath add"
37         @post (a + b < a || a + b < b) == __reverted
38         @post !__reverted -> c == a + b
39         @post !__reverted -> !__has_overflow
40         @post !(__has_buf_overflow)
41     */
```

Line 42-48 in File SwipeWallet.sol


```
42     function add(uint a, uint b) internal pure returns (uint c) {
43
44         c = a + b;
45
46         require(c >= a);
47
48     }
```

 The code meets the specification.

Formal Verification Request 2

SafeMath sub

 19, Aug 2019

 11.74 ms

Line 50-55 in File SwipeWallet.sol

```
50     /*@CTK "SafeMath sub"
51         @post (a < b) == __reverted
52         @post !__reverted -> c == a - b
53         @post !__reverted -> !__has_overflow
54         @post !(__has_buf_overflow)
55     */
```

Line 56-62 in File SwipeWallet.sol


```
56     function sub(uint a, uint b) internal pure returns (uint c) {
57
58         require(b <= a);
59
60         c = a - b;
61
62     }
```

 The code meets the specification.

Formal Verification Request 3

SafeMath mul

 19, Aug 2019

 98.77 ms

Line 64-69 in File SwipeWallet.sol

```
64  /*@CTK "SafeMath mul"
65     @post (((a) > (0)) && (((a) * (b)) / (a)) != (b))) == (__reverted)
66     @post !__reverted -> c == a * b
67     @post !__reverted == !__has_overflow
68     @post !(__has_buf_overflow)
69  */
```

Line 70-76 in File SwipeWallet.sol


```
70  function mul(uint a, uint b) internal pure returns (uint c) {
71
72      c = a * b;
73
74      require(a == 0 || c / a == b);
75
76  }
```

 The code meets the specification.

Formal Verification Request 4

SafeMath div

 19, Aug 2019

 11.06 ms

Line 78-84 in File SwipeWallet.sol

```
78  /*@CTK "SafeMath div"
79     @post (b == 0) == __reverted
80     @post !__reverted -> c == a / b
81     @post !__reverted -> !__has_overflow
82     @post !__reverted -> !__has_assertion_failure
83     @post !(__has_buf_overflow)
84  */
```

Line 85-91 in File SwipeWallet.sol


```
85  function div(uint a, uint b) internal pure returns (uint c) {
86
87      require(b > 0);
88
89      c = a / b;
90
91  }
```

 The code meets the specification.

Formal Verification Request 5

Owned

 19, Aug 2019

 8.16 ms

Line 159-161 in File SwipeWallet.sol

```
159  /*@CTK Owned
160      @post __post.owner == msg.sender
161  */
```

Line 162-166 in File SwipeWallet.sol


```
162  constructor() public {
163
164      owner = msg.sender;
165
166  }
```

 The code meets the specification.

Formal Verification Request 6

transferOwnership

 19, Aug 2019

 11.71 ms

Line 178-182 in File SwipeWallet.sol

```
178  /*@CTK transferOwnership
179      @tag assume_completion
180      @post owner == msg.sender
181      @post __post.owner == newOwner
182  */
```

Line 183-188 in File SwipeWallet.sol


```
183  function transferOwnership(address newOwner) public onlyOwner {
184
185      owner = newOwner;
186      emit OwnershipTransferred(owner, newOwner);
187
188  }
```

 The code meets the specification.

Formal Verification Request 7

freeze

 19, Aug 2019

 11.97 ms

Line 209-213 in File SwipeWallet.sol

```
209  /*@CTK freeze
210      @tag assume_completion
211      @post owner == msg.sender
212      @post __post.isLocked == 1
213  */
```

Line 214-218 in File SwipeWallet.sol


```
214  function freeze() public onlyOwner {
215      isLocked = 1;
216
217      emit Freezed();
218  }
```

✓ The code meets the specification.

Formal Verification Request 8

unfreeze

 19, Aug 2019

 13.23 ms

Line 220-223 in File SwipeWallet.sol

```
220  /*@CTK unfreeze
221      @tag assume_completion
222      @post __post.isLocked == 0
223  */
```

Line 224-228 in File SwipeWallet.sol


```
224  function unfreeze() public onlyOwner {
225      isLocked = 0;
226
227      emit UnFreezed();
228  }
```

✓ The code meets the specification.

Formal Verification Request 9

lockUser

 19, Aug 2019

 13.89 ms

Line 248-252 in File SwipeWallet.sol

```
248  /*@CTK lockUser
249      @tag assume_completion
250      @post owner == msg.sender
251      @post __post.blacklist[who]
252  */
```

Line 253-257 in File SwipeWallet.sol


```
253     function lockUser(address who) public onlyOwner {
254         blacklist[who] = true;
255
256         emit LockUser(who);
257     }
```

✓ The code meets the specification.

Formal Verification Request 10

unlockUser

 19, Aug 2019

 13.89 ms

Line 259-263 in File SwipeWallet.sol

```
259     /*@CTK unlockUser
260         @tag assume_completion
261         @post owner == msg.sender
262         @post !__post.blacklist[who]
263     */
```

Line 264-268 in File SwipeWallet.sol


```
264     function unlockUser(address who) public onlyOwner {
265         blacklist[who] = false;
266
267         emit UnlockUser(who);
268     }
```

✓ The code meets the specification.

Formal Verification Request 11

SXP

 19, Aug 2019

 24.37 ms

Line 306-311 in File SwipeWallet.sol

```
306     /*@CTK SXP
307         @post __post.symbol == "SXP"
308         @post __post.name == "Swipe"
309         @post __post.decimals == 18
310         @post __post.balances[owner] == __post._totalSupply
311     */
```

Line 312-326 in File SwipeWallet.sol

```
312     constructor() public {
313
314         symbol = "SXP";
315
316         name = "Swipe";
317     }
```


```
318     decimals = 18;
319
320     _totalSupply = 300000000 * 10**uint(decimals);
321
322     balances[owner] = _totalSupply;
323
324     emit Transfer(address(0), owner, _totalSupply);
325
326 }
```

✓ The code meets the specification.

Formal Verification Request 12

totalSupply

 19, Aug 2019

 28.08 ms

Line 336-339 in File SwipeWallet.sol

```
336     /*@CTK totalSupply
337         @tag assume_completion
338         @post __return == _totalSupply - balances[address(0)]
339     */
```

Line 340-344 in File SwipeWallet.sol


```
340     function totalSupply() public view returns (uint) {
341
342         return _totalSupply.sub(balances[address(0)]);
343
344     }
```

✓ The code meets the specification.

Formal Verification Request 13

balanceOf

 19, Aug 2019

 5.01 ms

Line 353-355 in File SwipeWallet.sol

```
353     /*@CTK balanceOf
354         @post balance == balances[tokenOwner]
355     */
```

Line 356-360 in File SwipeWallet.sol


```
356     function balanceOf(address tokenOwner) public view returns (uint balance) {
357
358         return balances[tokenOwner];
359
360     }
```

✓ The code meets the specification.

Formal Verification Request 14

transfer

 19, Aug 2019

 184.23 ms

Line 373-380 in File SwipeWallet.sol

```
373  /*@CTK transfer
374     @tag assume_completion
375     @pre msg.sender != to
376     @post isLocked == 0
377     @post !blacklist[msg.sender]
378     @post __post.balances[msg.sender] == balances[msg.sender] - tokens
379     @post __post.balances[to] == balances[to] + tokens
380  */
```

Line 381-391 in File SwipeWallet.sol


```
381  function transfer(address to, uint tokens) public validLock permissionCheck
382      returns (bool success) {
383      balances[msg.sender] = balances[msg.sender].sub(tokens);
384
385      balances[to] = balances[to].add(tokens);
386
387      emit Transfer(msg.sender, to, tokens);
388
389      return true;
390  }
391  }
```

 The code meets the specification.

Formal Verification Request 15

approve

 19, Aug 2019

 24.64 ms

Line 411-416 in File SwipeWallet.sol

```
411  /*@CTK approve
412     @tag assume_completion
413     @post isLocked == 0
414     @post !blacklist[msg.sender]
415     @post __post.allowed[msg.sender][spender] == tokens
416  */
```

Line 417-425 in File SwipeWallet.sol

```
417  function approve(address spender, uint tokens) public validLock permissionCheck
418      returns (bool success) {
419      allowed[msg.sender][spender] = tokens;
420
421      emit Approval(msg.sender, spender, tokens);
```

```
422
423     return true;
424
425 }
```

✓ The code meets the specification.

Formal Verification Request 16

transferFrom

📅 19, Aug 2019

🕒 290.35 ms

Line 447-455 in File SwipeWallet.sol

```
447  /*@CTK transferFrom
448     @tag assume_completion
449     @pre from != to
450     @post isLocked == 0
451     @post !blacklist[msg.sender]
452     @post __post.balances[from] == balances[from] - tokens
453     @post __post.balances[to] == balances[to] + tokens
454     @post __post.allowed[from][msg.sender] == allowed[from][msg.sender] - tokens
455 */
```

Line 456-468 in File SwipeWallet.sol

```
456  function transferFrom(address from, address to, uint tokens) public validLock
    permissionCheck returns (bool success) {
457
458      balances[from] = balances[from].sub(tokens);
459
460      allowed[from][msg.sender] = allowed[from][msg.sender].sub(tokens);
461
462      balances[to] = balances[to].add(tokens);
463
464      emit Transfer(from, to, tokens);
465
466      return true;
467
468  }
```

✓ The code meets the specification.

Formal Verification Request 17

allowance

📅 19, Aug 2019

🕒 4.55 ms

Line 480-482 in File SwipeWallet.sol

```
480  /*@CTK allowance
481     @post remaining == allowed[tokenOwner][spender]
482 */
```

Line 483-487 in File SwipeWallet.sol


```
483     function allowance(address tokenOwner, address spender) public view returns (uint
        remaining) {
484
485         return allowed[tokenOwner][spender];
486
487     }
```

✓ The code meets the specification.

Formal Verification Request 18

burn

 19, Aug 2019

 151.94 ms

Line 502-508 in File SwipeWallet.sol

```
502     /*@CTK burn
503         @tag assume_completion
504         @post isLocked == 0
505         @post !blacklist[msg.sender]
506         @post __post._totalSupply == _totalSupply - value
507         @post __post.balances[msg.sender] == balances[msg.sender] - value
508     */
```

Line 509-516 in File SwipeWallet.sol


```
509     function burn(uint256 value) public validLock permissionCheck returns (bool
        success) {
510         require(msg.sender != address(0), "ERC20: burn from the zero address");
511
512         _totalSupply = _totalSupply.sub(value);
513         balances[msg.sender] = balances[msg.sender].sub(value);
514         emit Transfer(msg.sender, address(0), value);
515         return true;
516     }
```

✓ The code meets the specification.

Formal Verification Request 19

approveAndCall

 19, Aug 2019

 25.08 ms

Line 527-532 in File SwipeWallet.sol

```
527     /*@CTK approveAndCall
528         @tag assume_completion
529         @post isLocked == 0
530         @post !blacklist[msg.sender]
531         @post __post.allowed[msg.sender][spender] == tokens
532     */
```

Line 533-545 in File SwipeWallet.sol

```

533     function approveAndCall(address spender, uint tokens, bytes memory data) public
        validLock permissionCheck returns (bool success) {
534
535         allowed[msg.sender][spender] = tokens;
536
537         emit Approval(msg.sender, spender, tokens);
538
539         ApproveAndCallFallBack(spender).receiveApproval(msg.sender, tokens, address(
            this), data);
540
541         return true;
542
543     }

```

✓ The code meets the specification.

Formal Verification Request 20

burnForAllowance

📅 19, Aug 2019

🕒 750.61 ms

Line 554-563 in File SwipeWallet.sol

```

554     /*@CTK burnForAllowance
555         @tag assume_completion
556         @pre account != feeAccount
557         @post owner == msg.sender
558         @post account != address(0)
559         @post balances[account] >= amount
560         @post __post.balances[account] == balances[account] - amount
561         @post __post.balances[feeAccount] == balances[feeAccount] + amount * 2 / 10
562         @post __post._totalSupply == _totalSupply - (amount - amount * 2 / 10)
563     */

```

Line 564-577 in File SwipeWallet.sol

```

564     function burnForAllowance(address account, address feeAccount, uint256 amount)
        public onlyOwner returns (bool success) {
565         require(account != address(0), "burn from the zero address");
566         require(balanceOf(account) >= amount, "insufficient balance");
567
568         uint feeAmount = amount.mul(2).div(10);
569         uint burnAmount = amount.sub(feeAmount);
570
571         _totalSupply = _totalSupply.sub(burnAmount);
572         balances[account] = balances[account].sub(amount);
573         balances[feeAccount] = balances[feeAccount].add(feeAmount);
574         emit Transfer(account, address(0), burnAmount);
575         emit Transfer(account, msg.sender, feeAmount);
576         return true;
577     }

```

✓ The code meets the specification.

Source Code with CertiK Labels

File SwipeWallet.sol

```

1  pragma solidity ^0.5.0;
2
3
4  // -----
5
6  // 'SXP' 'Swipe' token contract
7
8  //
9
10 // Symbol      : SXP
11
12 // Name        : Swipe
13
14 // Total supply: 300,000,000.000000000000000000
15
16 // Decimals    : 18
17
18 // Website     : https://swipe.io
19
20
21 //
22
23
24 // -----
25
26
27
28 // -----
29
30 // Safe maths
31
32 // -----
33
34 library SafeMath {
35
36     /*@CTK "SafeMath add"
37         @post (a + b < a || a + b < b) == __reverted
38         @post !__reverted -> c == a + b
39         @post !__reverted -> !__has_overflow
40         @post !(__has_buf_overflow)
41     */
42     function add(uint a, uint b) internal pure returns (uint c) {
43
44         c = a + b;
45
46         require(c >= a);
47
48     }
49
50     /*@CTK "SafeMath sub"
51         @post (a < b) == __reverted
52         @post !__reverted -> c == a - b
53         @post !__reverted -> !__has_overflow
54         @post !(__has_buf_overflow)

```

```

55  */
56  function sub(uint a, uint b) internal pure returns (uint c) {
57
58      require(b <= a);
59
60      c = a - b;
61
62  }
63
64  /*@CTK "SafeMath mul"
65   @post (((a) > (0)) && (((a) * (b)) / (a)) != (b))) == (__reverted)
66   @post !__reverted -> c == a * b
67   @post !__reverted == !__has_overflow
68   @post !(__has_buf_overflow)
69  */
70  function mul(uint a, uint b) internal pure returns (uint c) {
71
72      c = a * b;
73
74      require(a == 0 || c / a == b);
75
76  }
77
78  /*@CTK "SafeMath div"
79   @post (b == 0) == __reverted
80   @post !__reverted -> c == a / b
81   @post !__reverted -> !__has_overflow
82   @post !__reverted -> !__has_assertion_failure
83   @post !(__has_buf_overflow)
84  */
85  function div(uint a, uint b) internal pure returns (uint c) {
86
87      require(b > 0);
88
89      c = a / b;
90
91  }
92
93 }
94
95
96
97 // -----
98
99 // ERC Token Standard #20 Interface
100
101 // https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md
102
103 // -----
104
105 contract ERC20Interface {
106
107     function totalSupply() public view returns (uint);
108
109     function balanceOf(address tokenOwner) public view returns (uint balance);
110
111     function allowance(address tokenOwner, address spender) public view returns (uint
        remaining);

```

```

112
113     function transfer(address to, uint tokens) public returns (bool success);
114
115     function approve(address spender, uint tokens) public returns (bool success);
116
117     function transferFrom(address from, address to, uint tokens) public returns (bool
        success);
118
119
120     event Transfer(address indexed from, address indexed to, uint tokens);
121
122     event Approval(address indexed tokenOwner, address indexed spender, uint tokens);
123
124 }
125
126
127
128 // -----
129
130 // Contract function to receive approval and execute function in one call
131
132 //
133
134 // Borrowed from MiniMeToken
135
136 // -----
137
138 contract ApproveAndCallFallBack {
139
140     function receiveApproval(address from, uint256 tokens, address token, bytes memory
        data) public;
141
142 }
143
144
145
146 // -----
147
148 // Owned contract
149
150 // -----
151
152 contract Owned {
153
154     address public owner;
155
156     event OwnershipTransferred(address indexed _from, address indexed _to);
157
158
159     /*@CTK Owned
160      @post __post.owner == msg.sender
161      */
162     constructor() public {
163
164         owner = msg.sender;
165
166     }
167

```

```

168
169     modifier onlyOwner {
170
171         require(msg.sender == owner);
172
173         _;
174
175     }
176
177
178     /*@CTK transferOwnership
179     @tag assume_completion
180     @post owner == msg.sender
181     @post __post.owner == newOwner
182     */
183     function transferOwnership(address newOwner) public onlyOwner {
184
185         owner = newOwner;
186         emit OwnershipTransferred(owner, newOwner);
187
188     }
189
190 }
191
192 // -----
193
194 // Tokenlock contract
195
196 // -----
197 contract Tokenlock is Owned {
198
199     uint8 isLocked = 0;    //flag indicates if token is locked
200
201     event Freezed();
202     event UnFreezed();
203
204     modifier validLock {
205         require(isLocked == 0);
206         _;
207     }
208
209     /*@CTK freeze
210     @tag assume_completion
211     @post owner == msg.sender
212     @post __post.isLocked == 1
213     */
214     function freeze() public onlyOwner {
215         isLocked = 1;
216
217         emit Freezed();
218     }
219
220     /*@CTK unfreeze
221     @tag assume_completion
222     @post __post.isLocked == 0
223     */
224     function unfreeze() public onlyOwner {
225         isLocked = 0;

```



```

226
227     emit UnFreezed();
228 }
229 }
230
231 // -----
232
233 // Limit users in blacklist
234
235 // -----
236 contract UserLock is Owned {
237
238     mapping(address => bool) blacklist;
239
240     event LockUser(address indexed who);
241     event UnlockUser(address indexed who);
242
243     modifier permissionCheck {
244         require(!blacklist[msg.sender]);
245         _;
246     }
247
248     /*@CTK lockUser
249         @tag assume_completion
250         @post owner == msg.sender
251         @post __post.blacklist[who]
252     */
253     function lockUser(address who) public onlyOwner {
254         blacklist[who] = true;
255
256         emit LockUser(who);
257     }
258
259     /*@CTK unlockUser
260         @tag assume_completion
261         @post owner == msg.sender
262         @post !__post.blacklist[who]
263     */
264     function unlockUser(address who) public onlyOwner {
265         blacklist[who] = false;
266
267         emit UnlockUser(who);
268     }
269 }
270
271
272 // -----
273
274 // ERC20 Token, with the addition of symbol, name and decimals and a
275
276 // fixed supply
277
278 // -----
279
280 contract SwipeToken is ERC20Interface, Tokenlock, UserLock {
281
282     using SafeMath for uint;
283

```

```
284
285     string public symbol;
286
287     string public name;
288
289     uint8 public decimals;
290
291     uint _totalSupply;
292
293
294     mapping(address => uint) balances;
295
296     mapping(address => mapping(address => uint)) allowed;
297
298
299
300     // -----
301
302     // Constructor
303
304     // -----
305
306     /*@CTK SXP
307         @post __post.symbol == "SXP"
308         @post __post.name == "Swipe"
309         @post __post.decimals == 18
310         @post __post.balances[owner] == __post._totalSupply
311     */
312     constructor() public {
313
314         symbol = "SXP";
315
316         name = "Swipe";
317
318         decimals = 18;
319
320         _totalSupply = 300000000 * 10**uint(decimals);
321
322         balances[owner] = _totalSupply;
323
324         emit Transfer(address(0), owner, _totalSupply);
325
326     }
327
328
329
330     // -----
331
332     // Total supply
333
334     // -----
335
336     /*@CTK totalSupply
337         @tag assume_completion
338         @post __return == _totalSupply - balances[address(0)]
339     */
340     function totalSupply() public view returns (uint) {
341
```

```

342     return _totalSupply.sub(balances[address(0)]);
343
344 }
345
346
347
348 // -----
349
350 // Get the token balance for account `tokenOwner`
351
352 // -----
353 /*@CTK balanceOf
354   @post balance == balances[tokenOwner]
355   */
356 function balanceOf(address tokenOwner) public view returns (uint balance) {
357
358     return balances[tokenOwner];
359
360 }
361
362
363
364 // -----
365
366 // Transfer the balance from token owner's account to `to` account
367
368 // - Owner's account must have sufficient balance to transfer
369
370 // - 0 value transfers are allowed
371
372 // -----
373 /*@CTK transfer
374   @tag assume_completion
375   @pre msg.sender != to
376   @post isLocked == 0
377   @post !blacklist[msg.sender]
378   @post __post.balances[msg.sender] == balances[msg.sender] - tokens
379   @post __post.balances[to] == balances[to] + tokens
380   */
381 function transfer(address to, uint tokens) public validLock permissionCheck
382     returns (bool success) {
383
384     balances[msg.sender] = balances[msg.sender].sub(tokens);
385
386     balances[to] = balances[to].add(tokens);
387
388     emit Transfer(msg.sender, to, tokens);
389
390     return true;
391
392 }
393
394
395 // -----
396
397 // Token owner can approve for `spender` to transferFrom(...) `tokens`
398

```

```

399 // from the token owner's account
400
401 //
402
403 // https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20-token-standard.md
404
405 // recommends that there are no checks for the approval double-spend attack
406
407 // as this should be implemented in user interfaces
408
409 // -----
410
411 /*@CTK approve
412   @tag assume_completion
413   @post isLocked == 0
414   @post !blacklist[msg.sender]
415   @post __post.allowed[msg.sender][spender] == tokens
416 */
417 function approve(address spender, uint tokens) public validLock permissionCheck
418   returns (bool success) {
419     allowed[msg.sender][spender] = tokens;
420
421     emit Approval(msg.sender, spender, tokens);
422
423     return true;
424 }
425
426
427
428
429 // -----
430
431 // Transfer `tokens` from the `from` account to the `to` account
432
433 //
434
435 // The calling account must already have sufficient tokens approve(...)-d
436
437 // for spending from the `from` account and
438
439 // - From account must have sufficient balance to transfer
440
441 // - Spender must have sufficient allowance to transfer
442
443 // - 0 value transfers are allowed
444
445 // -----
446
447 /*@CTK transferFrom
448   @tag assume_completion
449   @pre from != to
450   @post isLocked == 0
451   @post !blacklist[msg.sender]
452   @post __post.balances[from] == balances[from] - tokens
453   @post __post.balances[to] == balances[to] + tokens
454   @post __post.allowed[from][msg.sender] == allowed[from][msg.sender] - tokens
455 */

```

```

456 function transferFrom(address from, address to, uint tokens) public validLock
    permissionCheck returns (bool success) {
457
458     balances[from] = balances[from].sub(tokens);
459
460     allowed[from][msg.sender] = allowed[from][msg.sender].sub(tokens);
461
462     balances[to] = balances[to].add(tokens);
463
464     emit Transfer(from, to, tokens);
465
466     return true;
467 }
468
469
470
471
472 // -----
473
474 // Returns the amount of tokens approved by the owner that can be
475
476 // transferred to the spender's account
477
478 // -----
479
480 /*@CTK allowance
481   @post remaining == allowed[tokenOwner][spender]
482   */
483 function allowance(address tokenOwner, address spender) public view returns (uint
    remaining) {
484
485     return allowed[tokenOwner][spender];
486
487 }
488
489
490 // -----
491 // Destroys `amount` tokens from `account`, reducing the
492 // total supply.
493
494 // Emits a `Transfer` event with `to` set to the zero address.
495
496 // Requirements
497
498 // - `account` cannot be the zero address.
499 // - `account` must have at least `amount` tokens.
500
501 // -----
502 /*@CTK burn
503   @tag assume_completion
504   @post isLocked == 0
505   @post !blacklist[msg.sender]
506   @post __post._totalSupply == _totalSupply - value
507   @post __post.balances[msg.sender] == balances[msg.sender] - value
508   */
509 function burn(uint256 value) public validLock permissionCheck returns (bool
    success) {
510     require(msg.sender != address(0), "ERC20: burn from the zero address");

```

```

511     _totalSupply = _totalSupply.sub(value);
512     balances[msg.sender] = balances[msg.sender].sub(value);
513     emit Transfer(msg.sender, address(0), value);
514     return true;
515 }
516
517 // -----
518 // Token owner can approve for `spender` to transferFrom(...) `tokens`
519 // from the token owner's account. The `spender` contract function
520 // `receiveApproval(...)` is then executed
521 // -----
522 /*@CTK approveAndCall
523   @tag assume_completion
524   @post isLocked == 0
525   @post !blacklist[msg.sender]
526   @post __post.allowed[msg.sender][spender] == tokens
527 */
528 function approveAndCall(address spender, uint tokens, bytes memory data) public
529   validLock permissionCheck returns (bool success) {
530
531     allowed[msg.sender][spender] = tokens;
532
533     emit Approval(msg.sender, spender, tokens);
534
535     ApproveAndCallFallBack(spender).receiveApproval(msg.sender, tokens, address(
536       this), data);
537
538     return true;
539 }
540
541 // -----
542 // Destroys `amount` tokens from `account`. `amount` is then deducted
543 // from the caller's allowance.
544 // See `burn` and `approve`.
545 // -----
546 /*@CTK burnForAllowance
547   @tag assume_completion
548   @pre account != feeAccount
549   @post owner == msg.sender
550   @post account != address(0)
551   @post balances[account] >= amount
552   @post __post.balances[account] == balances[account] - amount
553   @post __post.balances[feeAccount] == balances[feeAccount] + amount * 2 / 10
554   @post __post._totalSupply == _totalSupply - (amount - amount * 2 / 10)
555 */
556 function burnForAllowance(address account, address feeAccount, uint256 amount)
557   public onlyOwner returns (bool success) {
558   require(account != address(0), "burn from the zero address");
559   require(balanceOf(account) >= amount, "insufficient balance");
560 }

```

```

566     uint feeAmount = amount.mul(2).div(10);
567     uint burnAmount = amount.sub(feeAmount);
568
569     _totalSupply = _totalSupply.sub(burnAmount);
570     balances[account] = balances[account].sub(amount);
571     balances[feeAccount] = balances[feeAccount].add(feeAmount);
572     emit Transfer(account, address(0), burnAmount);
573     emit Transfer(account, msg.sender, feeAmount);
574     return true;
575 }
576
577
578 // -----
579 // Don't accept ETH
580
581 // -----
582
583 function () external payable {
584
585     revert();
586
587 }
588
589
590
591
592 // -----
593
594 // Owner can transfer out any accidentally sent ERC20 tokens
595
596 // -----
597
598 function transferAnyERC20Token(address tokenAddress, uint tokens) public onlyOwner
599     returns (bool success) {
600
601     return ERC20Interface(tokenAddress).transfer(owner, tokens);
602
603 }
604 }

```