



			Primary-		Sub- (i ≥ 1)	
			Private	Public	Public	Private
OFF-CHAIN (payee's data)	Spend keys		Common key derivation method 		$S_i = H_s(\text{"SubAddr" } v_0 i) G + S_0$	$S_i = H_s(\text{"SubAddr" } v_0 i) + S_0$
	View keys		$V_0 = v_0 G$		$V_i = v_0 S_i$	$V_i = v_0 S_i$ <i>Actually never used: subaddresses have been designed to share v₀ usage (for blockchain-scanning performance reasons)</i>
	Addresses		$\text{Base58}(0x12 S_0 V_0 \text{checksum}) = \text{"4"} \quad [95 \text{ chars}]$ 4 bytes-truncated Keccak256 hash		$\text{Base58}(0x2A S_i V_i \text{checksum}) = \text{"8"} \quad [95 \text{ chars}]$ 4 bytes-truncated Keccak256 hash	
	Integrated addresses		8 byte-compact paymentID, encrypted in paying transaction (vs 32 byte former one) $\text{Base58}(0x13 S_0 V_0 \text{payID} \text{checksum}) = \text{"4"} \quad [106 \text{ chars}]$ 4 bytes-truncated Keccak256 hash		N/A because integrated addresses and sub-addresses solve somewhat the same problem From https://monerodocs.org/public-address/integrated-address/ : "[...] Individuals should prefer subaddresses to receive payments. This is to improve privacy in certain scenarios. See article on subaddresses for details. Businesses accepting payments in an automated way should prefer integrated addresses . The rationale is as follows: [...]"	
ON-CHAIN (by payer's initiative)	Transaction keys		$R = r G$		$R = r S_i$	r
	Stealth Addresses (t ≥ 0)	Payer's POV	From payer's POV, the private key isn't known because the address we are dealing with is the transaction destination, the recipient going to be paid by the payer $X_t = H_s(r V_0 t) G + S_0$		$X_t = H_s(r V_i t) G + S_i$	From payer's POV, the private key isn't known because the address we are dealing with is the transaction destination, the recipient going to be paid by the payer
		Payee's POV	Used for Ring Signatures when payee will in turn become payer spending this UT XO $X_t = H_s(v_0 R t) + S_0$	$X_t = H_s(v_0 R t) G + S_0$	$X_t = H_s(v_0 R t) G + S_i$	Used for Ring Signatures when payee will in turn become payer spending this UT XO $X_t = H_s(v_0 R t) + S_i$
"Elliptic notes" ☺		Lower case letters and H _s outputs are scalar values. UPPER case letters denote points on Monero-chosen elliptic curve (twisted Edwards Ed25519), even if they can be represented as a single 256 bit value thanks to a technique known as compression (representation used in hashing after relevant EC algebra has been applied, in addresses, in protocol fields). So, when involving EC points, products and sums have to be intended as their elliptic curves variant (acting on a 2D discrete space), not as usual scalar ones working on "compressed points values". H _s () = sc_reduce32(Keccak256()) : the Keccak hash output is capped by sc_reduce32() due to EC points' cyclic nature (special thanks to Koe for having pointed this out); note that the same constraint applies to the transaction private key r and to the cited 25-words-Mnemonic-phrase key derivation method (as well as to any other ones).				
Credits		Mastering Monero (First Edition - December 2018 / Free PDF - 18th April 2019 - SerHack and the Monero Community) Zero to Monero: Second Edition (v2.0.0 - April 4, 2020 - Koe, Kurt M. Alonso, Sarang Noether) chapters 1, 2, 4 Review of Cryptonote White Paper (July 2014 ? - Brandon Goodell AKA Surae Noether) How Cryptonote Addresses Are Created (luigi1111) Various topics from Monero Stack Exchange and Monerodocs NOTE: this cheatsheet's notation is slightly different from sources', trying to fit its "at first glance recap" function				